CIPHER SYSTEMS BASED ON CYCLIC DIFFERENCE SETS

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

M. SETHURAMAN

to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
FEBRUARY, 1986

Lovingly dedicated

to

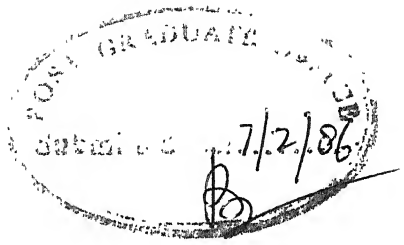
my wife Smt. Rama, kids Kumar & Menaka

who made this achievement possible

and to

my thesis supervisor Dr. M.U. Siddiqi

who made it worthwhile



CERTIFICATE

This is to certify that the thesis entitled "CIPHER SYSTEMS BASED ON CYCLIC DIFFERENCE SETS" has been carried out by Shri M. Sethuraman under my supervision and that it has not been submitted elsewhere for a degree.

(M.U. Siddiqi)
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Feb, 1986

ACKNOWLEDGEMENTS

I am pleased to have written my thesis in the warm environment provided by 'younger' M.Tech. batchmates. Many of them willingly contributed towards the thesis.

I am particularly thankful to the attention & encouragement received from my thesis supervisor Dr. M.U. Siddiqi. Many questions, raised by him during discussions, shaped the thesis into its present form.

I profited from many discussions with M/s. Vellaisamy, Somesh Kumar and L.S. Biradar at various stages of the thesis. They really took pains to clear some of my misconceptions.

My wife and kids deserve more than thanks. She devoted her energy to the family and insulated me completely from this responsibility of mine, so that I could pursue my degree. This thesis is as much theirs as it is mine.

I am especially grateful to Dr. R.P. Shenoy, Director, LRDE and Mr. V.G. Rao, Divisional Officer, B Communication Division, LRDE for sponsoring me to the course. I also thank Mr. R. Subramaniam and Mr. N. Sitaram who evinced keen interest on my thesis.

My heartfelt thanks is due to Mr. V.S. Mahalingam who willingly took care of my financial and administrative problem at Bangalore.

Finally I appreciate Mr. Raj Khanna for the excellent typing of the thesis, from an almost illegible manuscript.

M. SETHURAMAN

CONTENTS

		Page
List of Tables	...	iii
List of Figures	...	iv
Abstract	...	vi
Chapter 1. INTRODUCTION		
1.1 Scope of the Present Work	...	1
1.2 Organisation of the Thesis	...	2
Chapter 2. CRYPTOGRAPHIC SYSTEMS		
2.1 Scenerio	...	4
2.2 Cryptographic Systems Overview	...	6
2.2.1 Cryptanalytic Attacks	...	6
2.2.2 System Requirements	...	9
Chapter 3. BLOCK CIPHERS		
3.1 Perfect Secrecy	...	11
3.2 Block Ciphers using Symmetric Group S_N	...	13
CHAPTER 4. PERMUTATION NETWORKS		
4.1 Realisation through Degree Vector B..		20
4.2 Realisation through Cyclic Shifts ...		25
4.3 Realisation through Selection	...	27
Chapter 5. RESTRICTED-PERMUTATION NETWORKS		
5.1 Quasi-random Permutations	...	30
5.1.1 Permutation Selection Criteria	...	30
5.1.2. Permutation Tables with Good Correlation Properties...		32
5.2 Quasi-random Permutations through Binary Sequences of Length 2^n	...	34

Page

5.2.1 Binary Sequence Correlation... 35
5.2.2 Difference Set Concepts and Construction ... 37
5.2.3 Properties of m-sequences ... 40
5.2.4 Near-ideal 2^n Length Sequences ... 45
5.2.5 Acceptable 2^n Length Sequences ... 48
5.2.6 Generalisation of Results ... 61
5.2.7. Enumeration of Tables ... 62

Chapter 6. HARDWARE IMPLEMENTATION
6.1 Initial Approach ... 66
6.1.1 Encryption ... 66
6.1.2 Difficulty in Decryption ... 69
6.1.3 Solution through Involution... 71
6.2 Final Structure and its Capabilities. 72
6.2.1 Block Schematic ... 72
6.2.2 Permutation Space of the Cipher System ... 73
6.2.3 Cryptanalysis of the Final Scheme ... 77
6.3 Hardware and its Performance ... 79

Chapter 7. CONCLUSION
7.1 Results ... 88
7.2 Scope for Further Work ... 88

References ... 90

Appendix A1. List of Galois Field Elements and Difference Sets
A.1.1 $GF(2^5)$
A.1.2 $GF(2^6)$

LIST OF TABLES

Table No.		Page
5.1	(15,8,4)-Cyclic Difference Set Generated by GF(2^4) ...	41
5.2	Partition of Differences ...	42
5.3	Partial Correlation Results of 15 Length Sequence ...	46
5.4	Partition of Differences and Correlation of 2^4 Length Sequence ...	50
5.5	(2^4-1)-Design ...	52
5.6	2^4-Design extended from (2^4-1)-Design ...	54
5.7	2^5-Design through Primitive Polynomial $p(X) = 1 + X + X^2 + X^3 + X^5$ ...	56
5.8	2^5-Design through $p(X) = 1 + X + X^3 + X^4 + X^5$	57
5.9	2^5-Design through $p(X) = 1 + X^3 + X^5$ ...	58
5.10	2^5-Design through $p(X) = 1 + X^3 + X^5$ ...	59
5.11	2^6-Design through $p(X) = 1 + X^5 + X^6$ ...	60
5.12	Number of Permutation Tables ...	64
6.1	Log Table for GF(2^4) using $p(X) = 1 + X + X^4$	70
6.2	γ-Even Permutation ...	75
6.3	π_{fi}-Even Permutation ...	76
6.4	Read Muller Canonical Boolean Expression ...	78

LIST OF FIGURES

Fig. No.		Page
2.1	Privacy Homomorphism ...	5
2.2	Cryptographic System ...	9
3.1	Perfect Secrecy ...	12
3.2	A 3 bits Block Cipher ...	14
4.1	Permutation Network ...	21
4.2	Control Algorithm of Permutation Network ...	24
4.3	Block Cipher by Degree Vector B ...	24
4.4	Block Cipher (minimal Hardware) by Degree Vector B ...	25
4.5	Block Cipher by C Vector ...	27
4.6	Block Cipher by S Vector ...	28
5.1	Wire-Crossing Permutation ...	30
5.2	Cyclic Shift System ...	33
5.3	Dyadic Shift System ...	33
5.4	Near-ideal 2^n Design ...	55
5.5	'Acceptable' 2^n Design ...	62
6.1	General Sketch of the Cipher ...	66
6.2	Encryption Hardware Schematic ...	68
6.3	One Round of Cipher System : Product of Two Involutions π_{fi} and γ ...	73
6.4	Final Block Schematic of the Cipher System ...	74

Fig. No. Page

6.5	Addition Modulo 2^{32}	...	81
6.6	GF(2^4) Array Multiplier	...	82
6.7	Circuit Schematic	...	83
6.8	GF(2^{32}) Exponentiator Timing Diagram	...	84
6.9	I/O Timing Diagram	...	85
6.10	Pipeline Operation	...	86

ABSTRACT

Design of block cipher systems, that are cryptographically strong and are easily translated into hardware, is studied. Full permutation networks, realising all the symmetric group permutations S_N, are known to have the desired cryptographic strength. Three specific methods, i.e. degree vector representation, cyclic shift vector representation and selection vector representation of permutations, that lead to a mapping of integers in the range 0 to $(N!-1)$ to specific permutation of S_N are proposed. Implementation of these three schemes require excessive hardware.

Keeping in view the required key space size only, quasi-random permutation networks are studied. It is established that these could be constructed by the use of 2^n length 'white' sequences. Using the tools of cyclic difference sets, it is shown that acceptable 2^n length sequences, possessing near-ideal properties, are obtainable from (2^n-1) length near-ideal sequences. Construction procedure of these sequences leads to $GF(2^n)$ computation as the cipher transformation. A final scheme, cryptographically strong and having enough key space, is shown to be achievable using reasonable size hardware. The proposed system based on available MSI/LSI chips, is shown to support a throughput of 125 k bits for a low-power Schottky TTL version and 250 k bits for a Schottky TTL version.

CHAPTER 1

INTRODUCTION

Cipher systems protect data in storage as well as during transmission over communication networks [17][9]. Here data means any sequence of (0,1) i.e., any binary sequence that represent a source of information. The source of information could be text, speech, image etc. In case of text the binary sequence will be the standard ASCII (American Standard Code for information interchange) representation of the individual characters. It represents the digitised form of speech and image. A cipher transforms the data into a meaningless one to all but the intended receiver.

1.1 Scope of the Present Work

This present work is devoted to the study of cipher systems that can easily be implemented in hardware. Towards this goal, the requirements of a class of ciphers, i.e., Block Ciphers, are established. As the symmetric group S_N of permutations provides a good block cipher [17, pp.231-233], an attempt is made to realise it in hardware. This is shown to require exponential order hardware and a search is made towards ciphers using quasi-random permutations out of S_N. Making use of difference set concepts [25, pp 19-20] an efficient hardware realisable algorithmic design of cipher systems, based on $(2^n-1,$

2^{n-1}, 2^{n-2})-cyclic difference sets, is arrived at. This is converted into a design using existing MSI/LSI ICs and its performance in respect of hardware complexity and throughput is evaluated.

1.2 Organisation of the Thesis

Chapter 2 presents an overview of the need for a cipher system and the requirements of the cryptographic system in order to sustain the types of possible cryptanalytic attacks.

Chapter 3 focuses on block cipher systems. The symmetric group of permutations S_N is shown to possess all that is required of a cipher system.

Chapter 4 attempts three elegant ways of hardware implementation of S_N and the resulting hardware is proved to be far too nigh.

Chapter 5 gives an efficient algorithmic solution to realise a restricted-permutation cipher through cyclic difference sets.

Chapter 6 translates the algorithm into a proposal of cipher systems and presents a detailed design of the cipher using available MSI/LSI chips. The performance of the hardware ia also evaluated.

Chapter 7, the concluding chapter, summarises the results

achieved in this work. Certain related areas for further work are suggested. Further groundwork in these areas will lead to conclusive results on the strength and also the implementation aspects of the proposed cipher systems.

CHAPTER 2

CRYPTOGRAPHIC SYSTEMS

In this chapter we briefly review the need for the cryptographic systems in the context of computer communication networks and also the requirements that are to be met by these systems. Section 2.1 presents a vivid picture of the threat to secrecy and section 2.2 formulates the requirements of the system and the kind of cryptanalytic attacks the system will be subjected to, in the actual environment of its use.

2.1 Scenerio

Era of computer network is about to usher in India. The INDONET project is being vigorously pursued by the Computer Maintenance Corporation, which envisages pooling of all computing resources through a nationwide digital Network. Simultaneously an Integrated Switching and Digital network (ISDN) is being planned by Department of Education, to interconnect the computers of IITs and IISc through satellite link. Given such a network lot of information flow is going to take place and also a lot of information is going to be processed and stored at nodes of this network. Digital data transmitted on links and stored information are vulnerable to eavesdropping, without actual physical intrusion, through the very same network. As the data is in machine readable form, it is an easy job to routinely

monitor the traffic and to record and analyse, the ones in which the eavesdropper is interested, by the use of inexpensive hardware. Hence it becomes imperative to solve this threat and to put to good use the network resource.

As of now cryptography appears to provide the only practical method of protecting data. While it does not guarantee absolute security, it makes it more costly and risky to penetrate systems to obtain data in storage or during transmission [9],[17]. However, if data is to be protected while it is being processed, secrecy can be achieved only through privacy homomorphism [22]. The basic idea here is that encrypted data, after processing, should be the same as if the data were first deciphered, processed in plaintext and finally reenciphered (Fig.2.1).

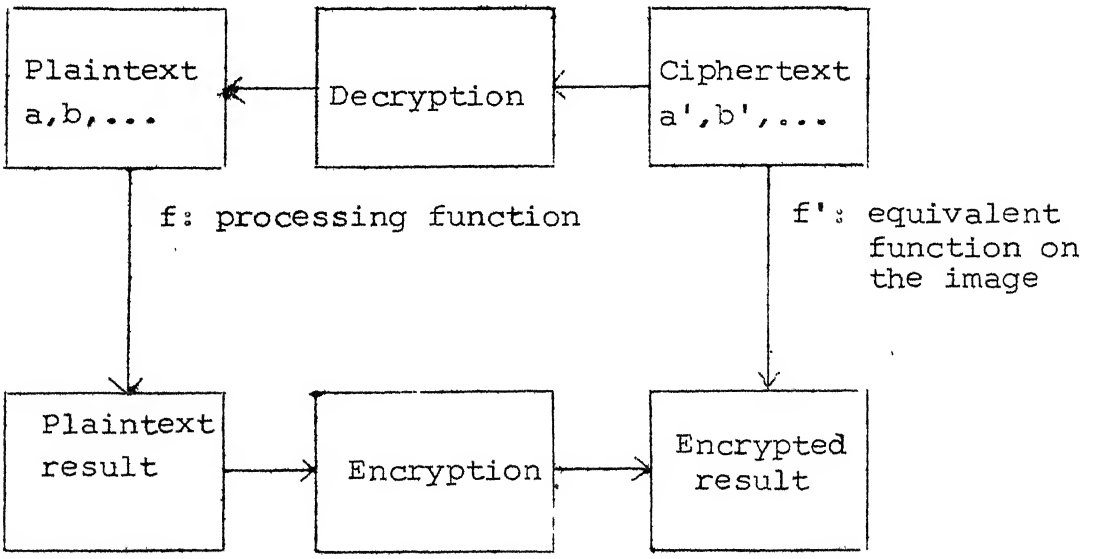


Fig. 2.1 Privacy Homomorphism

Though Privacy Homomorphisms offer the ultimate in data protection, it is not known whether it is possible to have a secure privacy homomorphism with a large number of operations [7, pp. 157-158],[22].

2.2 Cryptographic Systems Overview

Cryptography is the technique of protecting data. A cipher is a particular method of achieving secrecy and it protects data by transforming data from a usable and comprehensible 'plaintext' (message) form to a scrambled and incomprehensible 'ciphertext' (sometimes called cryptogram) form, from which the plaintext can only be recovered by use of a secret 'key'. The process of transforming plaintext into ciphertext is called 'encryption', the reverse process of transforming ciphertext into plaintext is called 'decryption'. Both encryption and decryption are controlled by cryptographic keys. The mechanism which performs these transformations, i.e. encryption and decryption, is known as a cryptographic system.

2.2.1 Cryptanalytic Attacks

Cryptanalysis is the science and study of methods of breaking ciphers [8],[9]. Cryptanalysis uses tools of probability theory and statistics, linear algebra, abstract algebra (group theory) and complexity theory. A cipher is 'breakable' if it is possible to determine the plaintext or key from the ciphertext, or to determine key from plaintext-ciphertext pairs. There are

two fundamentally different ways in which cryptographic systems may be secure. In some systems, the amount of information available to the cryptanalyst is actually insufficient to break the system, no matter how much computing power the cryptanalyst has at his disposal. A system of this kind is called 'unconditionally secure'. Even when the intercepted material contains sufficient information to allow a unique solution to the cryptanalytic problem, there is no guarantee that this solution can be found by a cryptanalyst with limited computational resources. It then becomes the goal of the designer of a cryptographic system to make encryption and decryption inexpensive, while ensuring that any successful cryptanalytic operation would be too complex to be economical. What is required is that the task of the cryptanalyst, though known to be achievable with a finite amount of computation, is so overwhelming as to exhaust the physical computing resources. We will call a task of this magnitude 'computationally infeasible' and the associated cryptographic system 'computationally secure'.

There are three basic methods of attack : cipher-text only, known-plaintext, and chosen-plaintext. In all cases it is assumed that the opponent knows the general system in use since this information can be obtained by studying a cryptographic device. Usually the worst circumstance from the point of view of cryptanalyst is to have nothing available to him but the

material he has intercepted and some knowledge of his opponent's messages. This may be limited to a knowledge of statistical properties of the language in use and a knowledge of certain probable words. This is the weakest threat to which a system is normally subjected, and any system which succumbs to it must be considered completely insecure. It is called a 'ciphertext only attack'.

Many secret messages sent for business purposes, press releases and product announcements, for example, are intended for subsequent public release. Hence a cryptanalyst often knows substantial amounts of corresponding plaintext and ciphertext, making it possible a 'known-plaintext attack'. While a known plaintext attack is not always possible, its occurrence is frequent enough that a system that succumbs to it is not considered secure.

The cryptanalyst is sometimes in the even stronger position of being able to see the ciphertext corresponding to any plaintext he chooses. His problem is to determine the key for later use in enchipering or deciphering other messages. This is 'chosen-plaintext attack'. For the purposes of certifying systems as secure, it is appropriate to consider more formidable cryptanalytic threats, as these not only give more realistic models of the working environment of a cryptographic system, but also make the assessment of the system's strength easier.

2.2.2 System Requirements

A cryptographic system has the following components [7, pp. 7-8] a plaintext message space (m), a ciphertext message space (c), a key space (k), a family of enciphering transformations (E_K : m $\longrightarrow$ c where K$\in$k), a family of deciphering transformations (D_K : c $\longrightarrow$ m, where K$\in$ k). Each encryption is defined by an encryption algorithm E, which is common to every transformation in the family, and a key K, which distinguishes it from the other transformations. Similarly, each decryption D_K is defined by a decryption algorithm D and a key K. For a given K, D_K is the inverse of E_K ; that is $D_K(E_K(M)) = M$ for every plaintext message M. See Fig.(2.2)

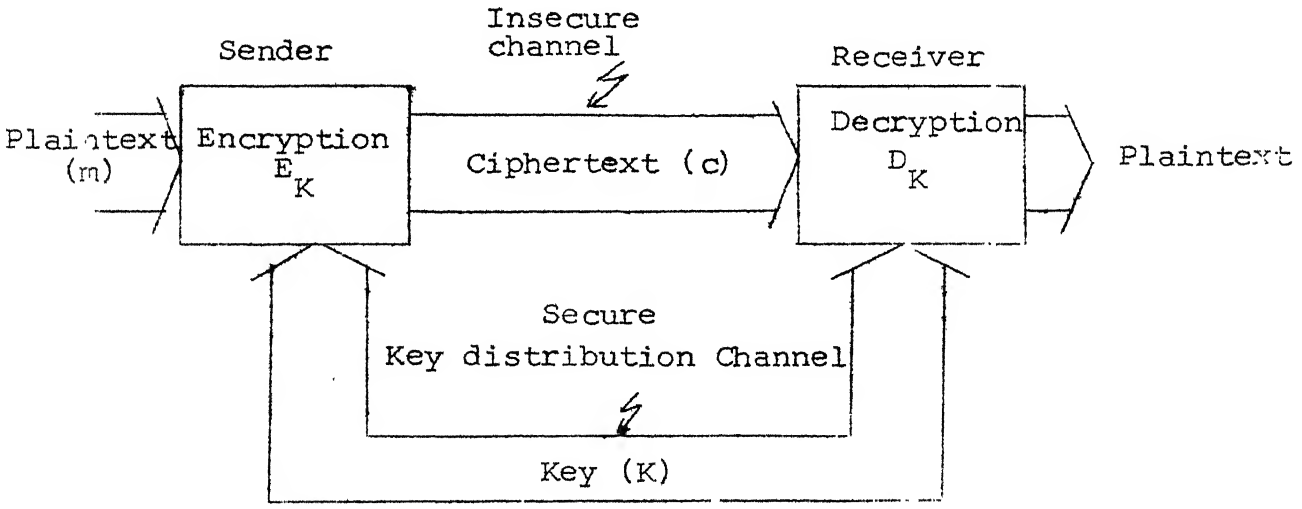


Fig. 2.2 Cryptographic System

There are specific requirements imposed on the cryptographic system to meet the secrecy aspects and also certain general requirements to meet its operational role in a network. These are

-It should be computationally infeasible for a cryptanalyst to systematically determine the deciphering transformation D_K even under the worst attack.

-It should be computationally infeasible for a cryptanalyst to systematically determine plaintext M from the intercepted ciphertext C.

-The security of the system should depend only on the secrecy of the keys and not on the secrecy of the algorithms E or D. The key space should be very large to prevent systematic key trial.

-It must be easy to use, less costly and should not degrade the network performance (speed, error etc). The transformations must be efficient and in general should not expand the massage length. An increase in length of text under encryption would unnecessarily complicate storage requirements and will reduce the effective rate of data transmission over communication links.

CHAPTER 3

BLOCK CIPHERS

In this chapter we focus our study on a class of ciphers called Block cipher. Section 3.1 postulates the 'perfect secrecy' requirement of such a cipher. Section 3.2 establishes that the symmetric group of all permutations S_N is a good block cipher.

3.1 Perfect Secrecy

Based on information theoretic properties of cryptographic systems Shannon [23] has proposed a theoretically secure system [7, p. 22]. The three classes of information in a cryptographic system are plaintext messages M occuring with probabilities $p(M)$ ($\sum_M p(M) = 1$), ciphertext messages C occuring with probabilities $p(C)$ ($\sum_C p(C) = 1$), keys K chosen with probabilities $p(K)$ where $\sum_K p(K) = 1$. 'Perfect secrecy' is defined by the condition $p(M/C) = p(M)$, where $p(M/C)$ is the probability that message M was sent given that C was received, that is, intercepting the ciphertext gives a cryptanalyst no additional information. If $p(C/M)$ is the conditional probability of receiving C given that M was sent, then $p(C/M)$ is the sum of the probabilities $p(K)$ of the keys K that encipher M as C i.e. $p(C/M) = \sum_{(E_K(M)=C)} p(K)$.

Usually there is atmost one key K such that $E_K(M) = C$ for given

M and C, but some ciphers can transform the same plaintext into the same ciphertext under different keys. A necessary & sufficient conditon for perfect secrecy is that for every C, p(C/M) = p(C) for all M. This means the probability of receiving a particular ciphertext C given that M was sent (enciphered under some key) is the same as the probability of receiving C given that some other message M was sent (enciphered under a different key). Perfect secrecy is possible using completely random keys at least as long as the messages they encipher. Fig. (3.1) illustrates a perfect secrecy system with four messages, all equally likely, and four keys, also equally likely. Here p(M/C) = p(M) = 1/4 and p(C/M) = p(C) = 1/4 for all M and C. A cryptanalyst intercepting one of the ciphertext messages C_1, C_2, C_3 or C_4 would have no way of determining which of the four keys was used and, therefore, whether the correct message is M_1, M_2, M_3 or M_4.

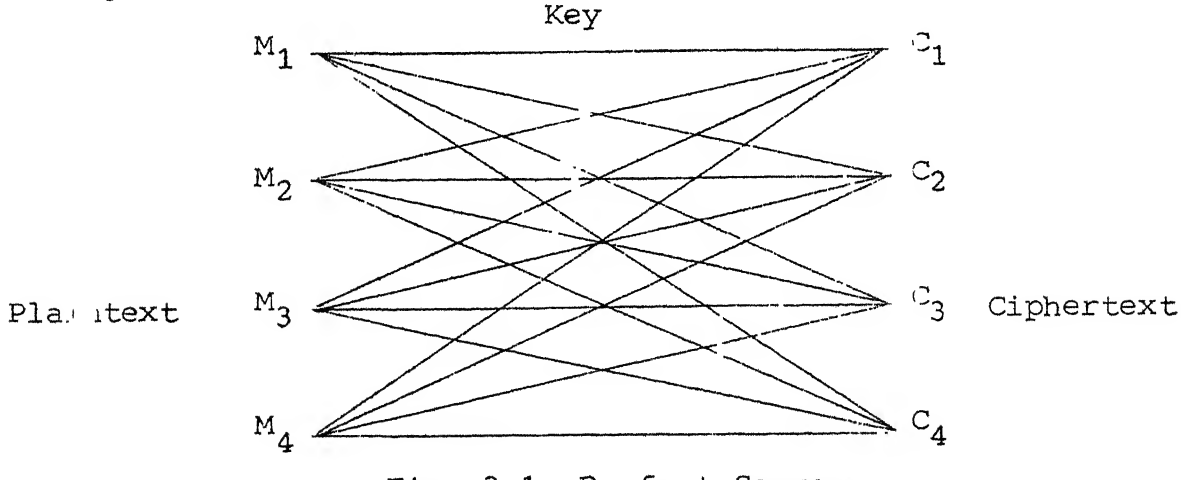


Fig. 3.1 Perfect Secrecy

Perfect secrecy requires that the number of keys must be at least as great as the number of possible messages. Otherwise there would be some message M such that for a given C, no K deciphers C into M, implying $p(M/C) = 0$. The cryptanalyst could thereby eliminate possible plaintext messages from consideration, increasing the chances of breaking the cipher.

3.2 Block Ciphers using Symmetric Group S_N

A block cipher is a cryptographic system which divides the plaintext into separate blocks, usually of same size, and operates on each independently to produce a sequence of ciphertext blocks. The most general possible block cipher is one which can transform any possible plaintext block into any possible ciphertext block as along as the overall transformation is reversible. This most naturally fits in the 'Perfect Secrecy' model explained earlier.

Let an n-block be a sequence of (0,1) of length n.

$$X = (X_o, X_1 \ldots X_{n-1}),\ x_i \in (0,1)$$

X can be interpretted as a vector V_n of length n, as well as the binary representation of the integer $0 \leq || x || \leq N-1$ where $N = 2^n$. A mapping $f : V_n \longrightarrow V_n$ can be represented by the two line notation

$$\begin{pmatrix} 1 , & 2 \ldots & N \\ i_1, & i_2 \ldots & i_N \end{pmatrix}$$

where the first line lists the elements in natural order and the second line lists the permuted order i.e.,the images of the elements under the mapping. In a general map the each image can take any one of the values 1 to N and hence there are N^N total maps. Out of these only the N! permutation maps are reversible i.e.,they are bijective maps. Here the images $i_1, i_2 \ldots i_N$ take on distinct values from 1 to N. These reversible mappings could be chosen as the cryptographic transformation. Hence an element of the symmetric group (S_N) of permutation could be chosen as the transformation. Even for moderate n = 16, the space of the transformations works out to a staggering figure of 2^{10^6} (one million bits). See Fig. 3.2 for a 3 bit block cipher.

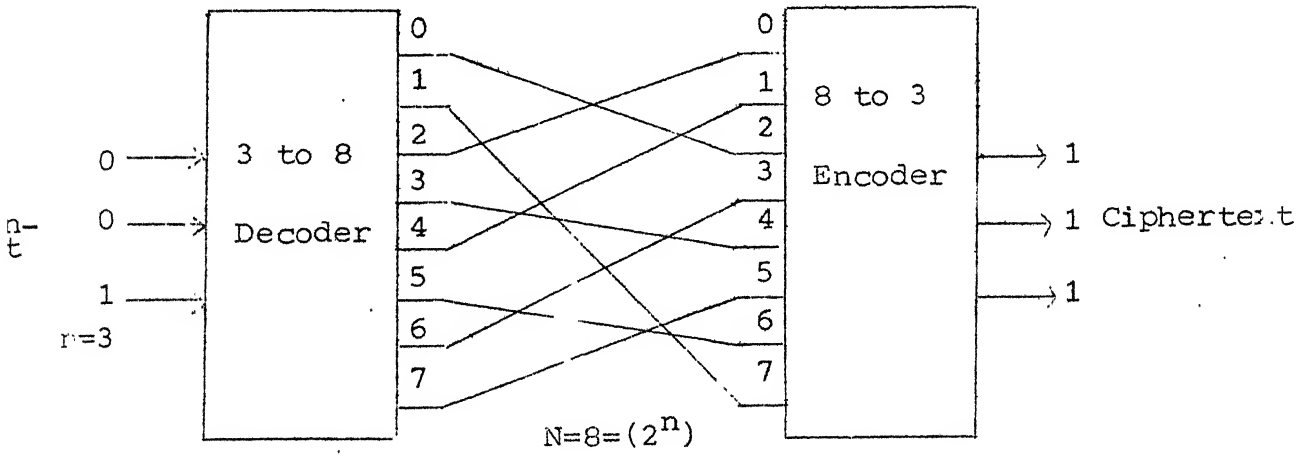


Fig. 3.2 A 3 bits Block Cipher

Using an element of S_N a keyed block cipher could be realised. User i and user j agree in some manner upon a key K in S_N and

transmit plaintext X enciphered by the permutation chosen ie $C = E_K(X)$. At the receiver the inverse permutation is performed to get the plaintext $X = D_K(C)$.

The block cipher using S_N fares very well in respect of all the three types of attack.

1. Ciphertext only attack : If reasonable length n is chosen as block length then it is very difficult to attempt ciphertext only attack as the side information such as frequence of occurrence & most probable word table etc is very difficult to generate on a size of $N = 2^n$. The size becomes 1.8×10^{19} even for $n = 64$. The key space of $(2^{64})!$ makes key trial absolutely impractical.

2. Known plaintext attack : An opponent who learns the correspondence between some subset of plaintext and ciphertext $X_i \longleftrightarrow C_i$, $0 \leq i < m$ can not on the basis of information alone find the plaintext corresponding to $y \notin (y_i)$. This still leaves $(N-m)!$ of $N!$ permutations in S_N satisfying the equations $E_K(X_i) = C_i$ $0 \leq i < m$. In order to uniquely fix K one needs to know the cryptograms of almost all the $N = 2^n$ members of the set. If the message symbols are equally likely then a staggering sequence of symbols are to be collected before one can expect all symbols to occur in the sequence.

In an experiment of drawing objects $X_1 \ldots X_N$, each with probability of occurrence $p(X_i) = 1/N$, one by one with replacement,

the expected length of sequence to contain all X_is atleast once can be found out as given below. Let N + z be the trial at which all N objects occur at least once for the first time i.e., in N+z-1 trials we have only N-1 objects occurring at least once and at $N+z^{th}$ trial N^{th} object occurs.

If we denote the occurrence of any distinct symbol for the first time by 1 and repetitions of previously occuring symbols by 0 then we have

$$\overbrace{101\ldots\ldots\ldots\ldots01}^{\text{Length} = N+z-1,\ \text{No. of ones} = N-1}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad}_{\text{total length } N+z}$$

No. of ways of having (N-1) ones in a length of N+z-1

$$= \binom{N+z-1}{N-1}$$

Probability of having (N-1) ones in a length of (N+z-1)

$$= \binom{N+z-1}{N-1} \left(1-\frac{1}{N}\right)^z \frac{1}{N^{N-1}}$$

where $\left(1-\frac{1}{N}\right)^z$ is probability of z failures

and $\frac{1}{N^{N-1}}$ is the probability of N-1 successes.

The probability of the experiment stopping at N+z

$$= \binom{N+z-1}{N-1} \left(1-\frac{1}{N}\right)^z \cdot \frac{1}{N^{N-1}} \cdot \frac{1}{N}$$

where $\frac{1}{N}$ is the probability of a success at the last draw.

$$P(Z=z) = \binom{N+z-1}{N-1} \left(1-\frac{1}{N}\right)^z \frac{1}{N^N} .$$

It can be verified that $\sum_{z=0}^{\infty} P(Z=z) = 1$.

$$\sum_{z=0}^{\infty} P(Z=z) = \sum_{z=0}^{\infty} \binom{N+z-1}{N-1} f^z s^N \quad \text{where } f = 1 - \frac{1}{N}$$

and $s = \frac{1}{N}$.

$$= s^N \sum_{z=0}^{\infty} \binom{N+z-1}{N-1} f^z$$

$$= s^N (1-f)^{-N} \qquad \text{by binomial series expansion}$$

$$= 1 \qquad \text{of } (1-f)^{-N}$$

Now the expected number of trials

$$E(Z) = \sum_{z=0}^{\infty} z \binom{N+z-1}{N-1} (1 - \frac{1}{N})^z \frac{1}{N^N}$$

$$= (N-1)N \sum_{z=1}^{\infty} z \binom{N+z-1}{N-1} \frac{1}{N} (1 - \frac{1}{N})^{z-1} \cdot \frac{1}{N^{N+1}}$$

$$= (N-1)N \sum_{z=1}^{\infty} \frac{(N+z-1)!}{(N-1)!\, z!} z \cdot \frac{1}{N} (1 - \frac{1}{N})^{z-1} \frac{1}{N^{N+1}}$$

$$= (N-1)N \sum_{z=1}^{\infty} \frac{(N+1+(z-1)-1)!}{(z-1)!\, N!} (1 - \frac{1}{N})^{z-1} \frac{1}{N^{N+1}}$$

$$= (N-1)N \sum_{y=0}^{\infty} \binom{M+y-1}{M-1} (1 - \frac{1}{N})^y (\frac{1}{N})^M$$

by letting $M = N+1$ & $y = z-1$.

$$= (N-1)N.$$

Hence the total length of the experiment is likely to be $N + (N-1)N = N^2$.

This would mean on the average a cryptanalyst has to

collect N^2 cryptograms. For n = 64 this works out to a staggering figure of $\simeq 4 \times 10^{38}$. Hence the cipher system is safe against known-plaintext attack.

3. Chosen-plaintext attack : Even here the cryptanalyst cannot hope to plant a message of length N.

So from all types of attack the block cipher using symmetric permutation group S_N is secure.

CHAPTER 4

PERMUTATION NETWORKS

Since the realisation of S_N provides a good cryptographic system, it will be nice if we can arrive at a hardware that can realise S_N. In this chapter, we describe in detail three elegant full permutation networks. Section 4.1 deals with realisation through degree vector representation of permutation. Section 4.2 presents cyclic shift vector realisation and finally section 4.3 gives an implementation of full permutation by selection of elements. In all these methods the complexity of hardware required is far too high.

For the purposes of a block cipher, it will be nice if all the permutations of S_N could be ordered in some way. After this any permutation in the set could be selected by specifying a random number within the range of 0 to $(N! - 1)$. If there exists a mapping which can be carried out algorithimically, then hardware realisation will be easier. We will look at possible implementations. Any permutation is specified by number x, $0 \leq x \leq (N! - 1)$ and this number is converted into a mixed radix digit system which uniquely specifies the position of permutation in the given ordering, in these implementations.

4.1 Realisation through Degree Vector B.

From any permutation $a_1a_2a_3\cdots a_{N-1}$ of S_{N-1}, one can derive N permutations of S_N by first placing N on the immediate right of a_{N-1} and then letting it step over a_{N-1}, $a_{N-2}\cdots a_1$ in turn [13, pp.332-338].

$a_1\ a_2\ \cdots\ a_{N-2}\ a_{N-1}\ a_N$

$a_1\ a_2\ \cdots\ a_{N-2}\ a_N\ a_{N-1}$

$a_N a_1\ a_2\ \cdots\ a_{N-2}\ a_{N-1}$

Starting from the unique permutation

1 of S_1, we get

12

21 the two permutations of S_2. Taking each of these in turn we get 123

132

312

213

231

321 the six permutations of S_3. Proceeding in this manner, taking each permutation in turn, we can derive from the $(N-1)!$ permutation of S_{N-1} all the $N!$ permutations of S_N.

The network in Fig. 4.1 is capable of realising S_N closely following the same way of generating permutations.

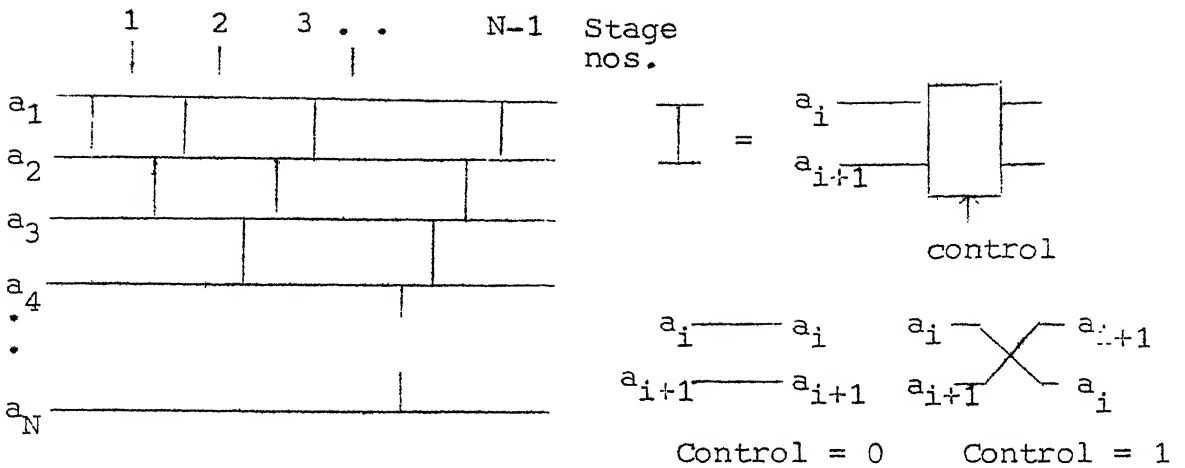


Fig. 4.1 Permutation Network

Stage 1 can produce S_2, stage 2 can insert a_3 in all three positions of S_2 creating S_3. For each of (N-1)! permutations obtained at $(N-2)^{th}$ stage, $(N-1)^{th}$ stage can insert a_N in all N positions of S_{N-1} creating S_N. However the hardware requires a total of $\frac{N(N-1)}{2}$ switching elements.

In a permuted version of elements $a_1 a_2 \ldots a_K \ldots a_N$ let b_K be the number of a's which are < K and are on its right side. Then the permutation is uniquely specified by its degree vector $B = B[1,2,\ldots,N] = [b_1 b_2 \ldots b_K \ldots b_N]$, $0 \leq b_K \leq K-1$. Degree vector representation gives a unique ordering of the permutations.

Example :

Position	Permutation	B vector
0	123	000
1	132	001
2	312	002
3	213	010
4	231	011
5	321	012

The position of a given permutation in the ordering can be obtained from the degree vector B.

Elements :	1	2	3	4 . . . N
B	b_1	b_2	b_3	$b_4 \cdots b_N$
	m_1	m_2	m_3	$m_3 \cdots m_N$

We start with $m_1 = b_1 = 0$, and then calculate the values of the other m's from the formula

$$m_K = K\, m_{K-1} + b_K \qquad K = 2,3,\ldots N.$$

In the algorithm, m_N gives the position of permutations in the ordering.

Therefore $m_N = b_N + Nb_{N-1} + N(N-1)b_{N-2} + \ldots + N!\, b_1$

$$= N! \sum_{K=1}^{N} b_K/K!$$

The justification for the algorithm is provided by the fact that the given permutation arises from a permutation of S_{N-1} obtainable from the given permutation by the deletion of N. If this permutation had m_{N-1} as its position in its own set, then the given permutation position is given by $Nm_{N-1} + b_N$ in the derived set, (eg) position of 3521746 in S_7

B[1,2,3,4,5,6,7] = [0120302]

Elements :	1	2	3	4	5	6	7
B :	0	1	2	0	3	0	2
m's	0	1	5	20	103	618	4328

The given permutation is 4328th in the set of 7! permutations of S_7. The position m_n of the permutation follows the mixed radix representation

$$m_N = N!\, b_1 + \ldots + \frac{N!\, b_K}{K!} + N(N-1)b_{N-2} + Nb_{N-1} + b_N$$

The digit weights are given by $w_i = \frac{N!}{i!}$ and $0 \leq b_K \leq K-1$.
The radices are $m_1 = 1$, $m_2 = 2$, ... $m_i = i \ldots m_N = N$.
(eg) Let us find out the degree vector B of 4328^{th} permutation.

$$4328 = 0 + (1 \times \frac{7!}{2!}) + (2 \times \frac{7!}{3!}) + (0 \times \frac{7!}{4!}) + (3 \times \frac{7!}{5!}) + (0 \times \frac{7!}{6!}) + (2 \times \frac{7!}{7!})$$

ie., $B = [0\ 1\ 2\ 0\ 3\ 0\ 2]$

Thus any number x $(0 \leq x \leq N! - 1)$ can be converted to a unique degree vector. Degree vector representation leads to a very simple control algorithm to set up the permutation network. The b_Ks of the degree vector indicates where the element K is to be inserted. At stage K switch off $(K-b_K)$ from top & switch on remaining b_K switches from bottom. See Fig. 4.2

The inverse permutation can be carried out by tracing the network in the reverse direction, using the same algorithm, ie., from output to input.

A block cipher realised using this type of permutation network is given below in Fig. 4.3.

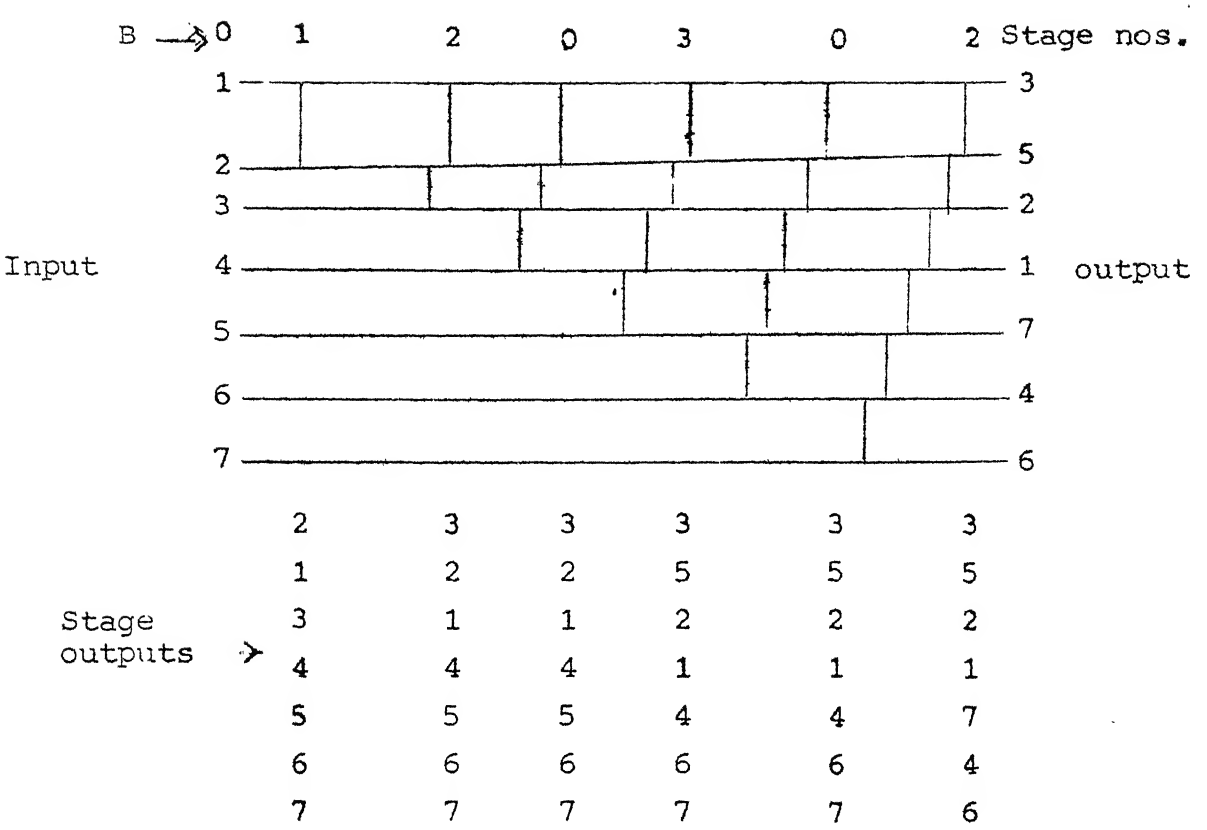


Fig. 4.2. Control Algorithm of Permutation Network

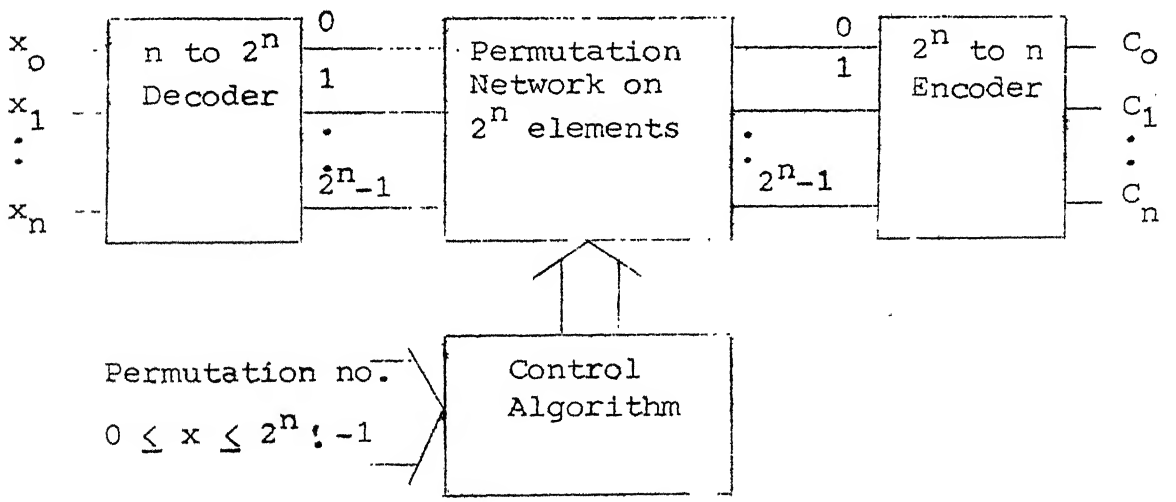


Fig. 4.3 Block Cipher by Degree Vector D.

In the block cipher realised above the permutation network can be set up in one step, however, it requires hardware proportional to $\frac{N(N-1)}{2}$. At the expense of delay we can realise the same using a hardware which is proportional to N, taking N steps to obtain the required permutation. A block cipher realised in this way is shown below in Fig. 4.4.

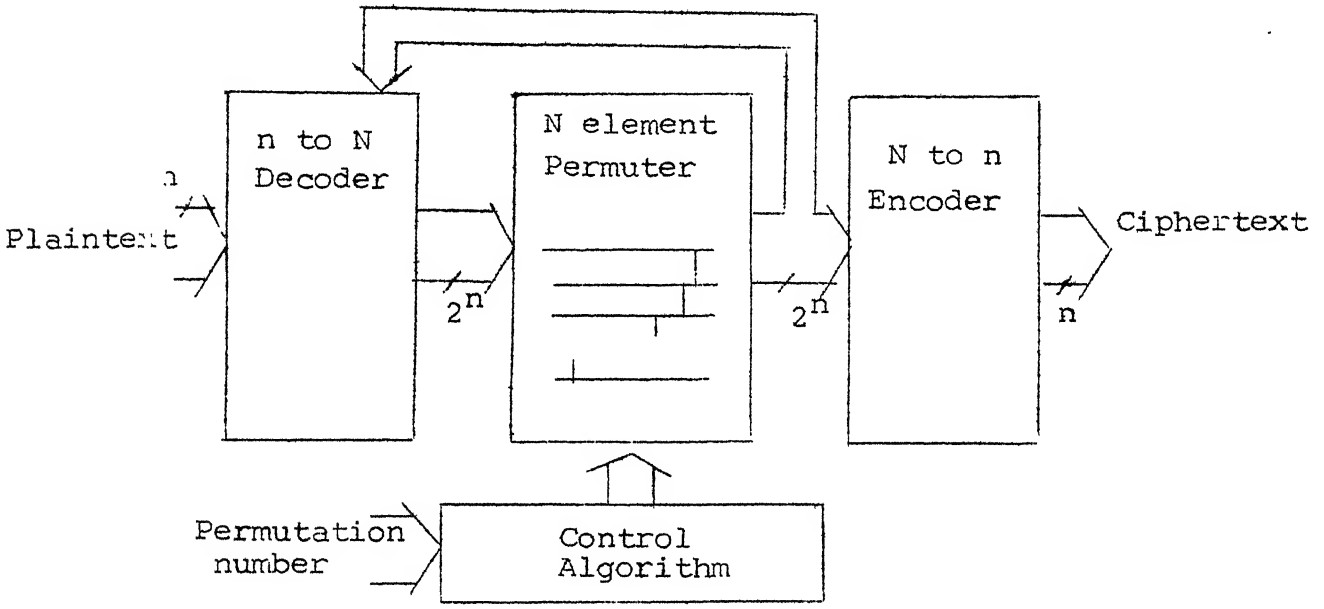


Fig. 4.4 Block Cipher (minimal hardware) by Degree Vector B

4.2. Realisation through Cyclic Shifts

From any permutation of $a_1a_2a_3\ldots a_{N-1}$ of S_{N-1}, one can derive N permutations of S_N by first placing ∂N on the immediate right of a_{N-1} and then taking cyclic shifts of the elements.

Initial order (0 shift)	$a_1\ a_2\ a_3 \cdots a_{N-1}\ a_N$
1 shift	$a_2\ a_3 \cdots a_{N-1}\ a_N\ a_1$
	.
	.
	.
(N-1) shifts	$a_N\ a_1 \cdots a_{N-1}$

Starting from the unique permutation

1 of S_1, we get

12

21 the two permutations of S_2, taking each of these in turn we get 123

231

312

213

132

321 the six permutations of S_3. Proceeding in this manner taking each permutation in turn, we can derive from the $(N-1)!$ permutations of S_{N-1} all the $N!$ permutations of S_N. Then any permutation of S_N can be uniquely represented by a cyclic shift vector $C = [C_1 C_2 \ldots C_K \ldots C_N]$

$0 \leq C_K \leq K-1$. The ordering according to cyclic shift vector is as follows.

Position	Permutation	C vector
0	123	000
1	231	001
2	312	002
3	213	010
4	132	011
5	321	012

Any number x $0 \leq x \leq N!-1$ can be uniquely converted into a C vector using mixed radix number system as described earlier.

A network can also be built which realises S_N using the C vector :

Fig. 4.5

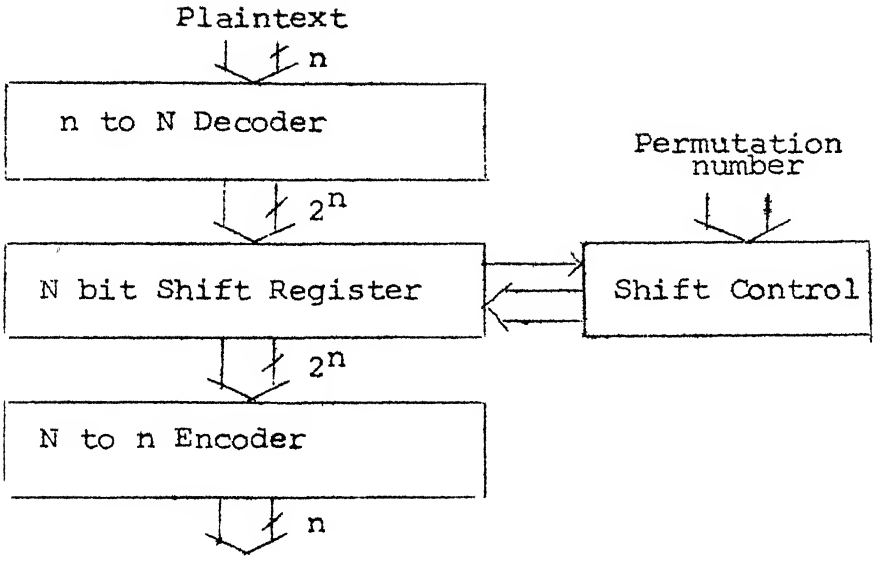


Fig. 4.5 Block Cipher by C Vector

4.3 Realisation through Selection

Following the definition of permutation, a permutation in S_N can be realised through a series of selection process for the elements in the successive positions, i.e., first element could be selected from among all the N elements, the next can be selected from the remaining N-1 elements and so on. Hence the permutation can be uniquely specified by a mixed radix number

$S = [S_1 \ldots S_K \ldots S_N]$.

With digit weights $w_i = \frac{N!}{i!}$ and $0 \leq S_K \leq K-1$.

This process can be mechanised through the following network (Fig. 4.6.)

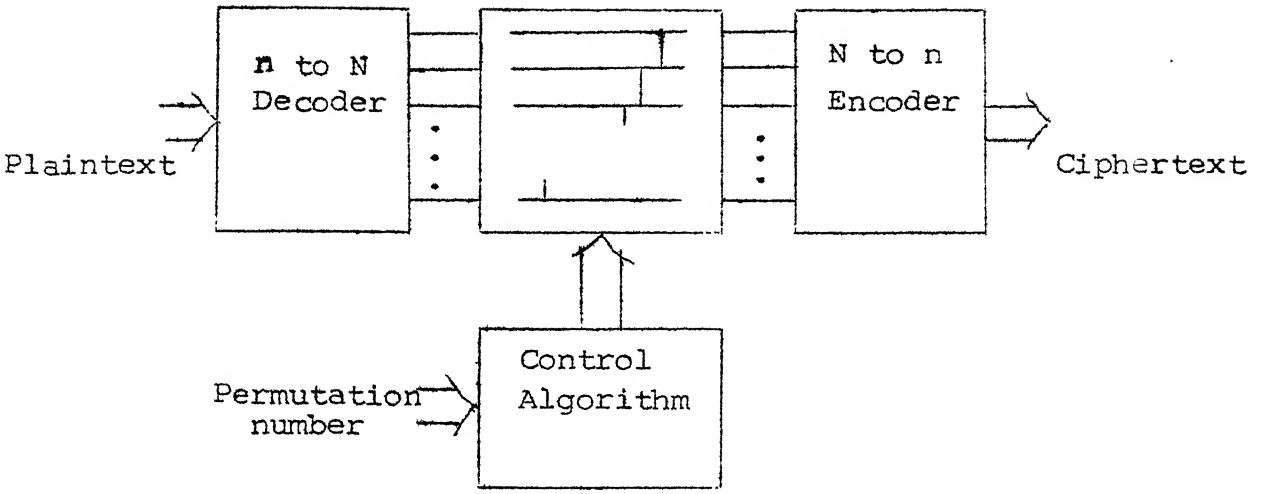


Fig. 4.6 Block Cipher by S Vector

The realisation of full permutation network requires a hardware complexity of the order of $N=2^n$, as seen from the above implementations.

CHAPTER 5

RESTRICTED-PERMUTATION NETWORKS

A keyed block cipher system, with a large enough block size n and capable of performing a random permutation from S_N, is effectively unbreakable due to impractical computational requirements of cryptanalysis. However we found that the hardware requirements to implement S_N is also too high. Added to this the length of binary digits required to specify the chosen random permutation is also far too high even for reasonable block lengths 'n' i.e. $\log_2(N!) \simeq (N-1).n$ bits, where $N=2^n$. A minimum key space of 2^n provides perfect secrecy and hence a less complex hardware configuration, which is capable of supporting a large enough ($\geq 2^n$) permutations out of S_N, will meet the requirements of the block cipher.

In this chapter we deal with methods of constructing restricted-permutation networks. Section 5.1 establishes the selection criteria (section 5.1.1) for the restricted permutation and the possibility of a compact construction through a good correlation single permutation table (section 5.1.2). Section 5.2 translates the problem into that of constructing 2^n length 'white' sequences and finds ways of constructing such sequences. Section 5.2.1 and 5.2.2 introduce concepts of binary sequence correlation and difference sets respectively. Section 5.2.3 reviews

the m-sequence properties. Section 5.2.4 postulates the theoretically obtainable near-ideal 2^n length sequence. Section 5.2.5 offers an 'acceptable' 2^n length sequences design through extension of 2^n-1 length sequences. After generalising the results in section 5.2.6, the enumeration of the number of permutation tables is given in section 5.2.7.

5.1 Quasi-random Permutations

5.1.1 Permutation Selection Criteria.

We could choose a fixed set of say M permutations in advance and store them. When a permutation is called for, we pick a number i at random between 1 and M and supply the i-th permutation. This may be called a quasi-random method [24] and has certain advantages over the truly random methods of choosing a permutation from S_N. With this algorithm one can weed out those permutations which do a poor job of scrambling or those which do not require great cryptanalytic effort. The wire-crossing permutation (see Fig. 5.1) is very weak from cryptanalytic point of view.

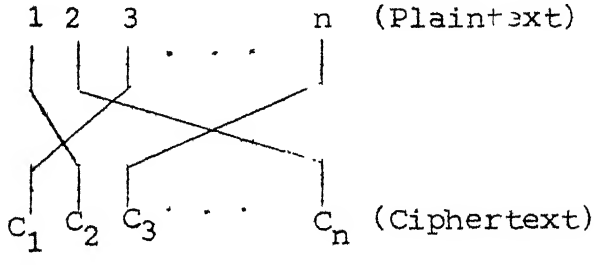


Fig. 5.1 Wire-Crossing Permutation

Wire crossing permutations are completely specified by a permutation matrix of size nxn and can be cracked if solution for the 'n' independent basis vectors are known. Whereas a block cipher substitution is specified by $2^n x 2^n$ permutation matrix, requiring solution for 2^n vectors. Wire crossing permutations account for n! permutations of S_N. Even those permutations which are linear and affine provide a weak cipher. They can be specified by again a (0,1) matrix of size nxn. Linear permutations are of the form C = Ax, A-nxn binary matrix. Affine permutations are of the form C = Ax + B where B is a n bit binary vector. The rows of A must be independent for reversibility, thus

row 1 of A may be chosen from $2^n - 2^0$ values

row 2 of A may be chosen from $2^n - 2^1$ values

row 3 of A may be chosen from $2^n - 2^2$ values

and so on until

row n of A may be chosen from $2^n - 2^{n-1}$ values

ie., the 1st row must be non zero, the 2nd must be non zero and distinct from 1st, the 3rd must be non zero and not any of the 4 linear functions of earlier ones and so on. Also B may be chosen in 2^n ways. Hence there are a total of $2^n(2^n-2^0)(2^n-2^1)$ $....(2^n-2^{n-1})$ nonsingular linear and affine transformations [12]. Cryptanalytic effort required to crack these require solutions for 'n' independent basis vectors only. In a truly random

selection of permutation from S_N there is always a chance, however small, that these poor permutation will be produced. In quasi-random method these can be entirely avoided.

5.1.2 Permutation Tables with Good Correlation Properties

Random permutation tables can be constructed by performing an experiment in which each member is chosen randomly one after the other without replacement, from an urn containing all the members. Whether the resultant permutation is random or whether the experiment was fair can be evaluated by statistical methods. For example run test which measures statistic of length of monotone subsequences i.e., 'runs up' segments that are increasing or 'runs down' segments that are decreasing or serial correlation test which measures the amount of dependance of U_{j+K} on U_j in sequence $U_o\ U_1 \ldots U_j \ldots U_{N-1}$. However the number of such tables required for a block cipher is very large and each permutation table requires a storage of $nx2^n$ bits. This is impractical from implementation point of view.

The requirement for the large number of tables could be reduced if we could locate a set of permutation tables whose shifted versions could also be used i.e., either cyclic shifts or dyadic shifts. To meet this requirement the sequence of each table should possess good randomness property for all its shifts Suppose we could locate one random permutation table of length N, which has good correlation property for all N shifts, then we

can use that single table with a key space which gives all its shifts. See Fig. 5.2

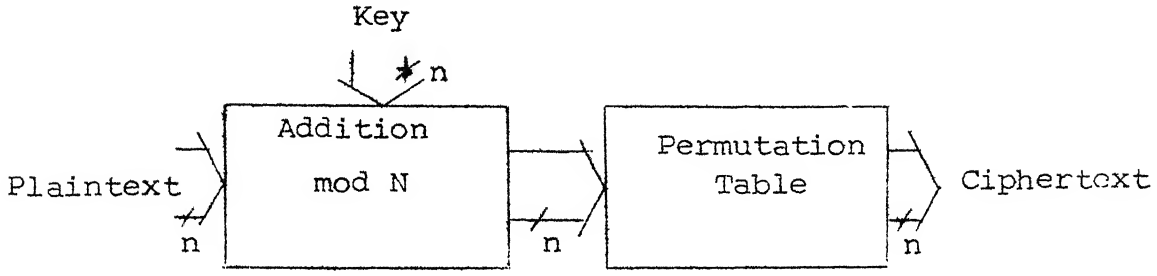


Fig. 5.2 Cyclic Shift System

Same way, if a permutation table could be constructed which has good correlation property for all its dyadic shift then key ('n' bits wide) could be Ex-OR mixed with plaintext ('n' bits wide). See Fig. 5.3

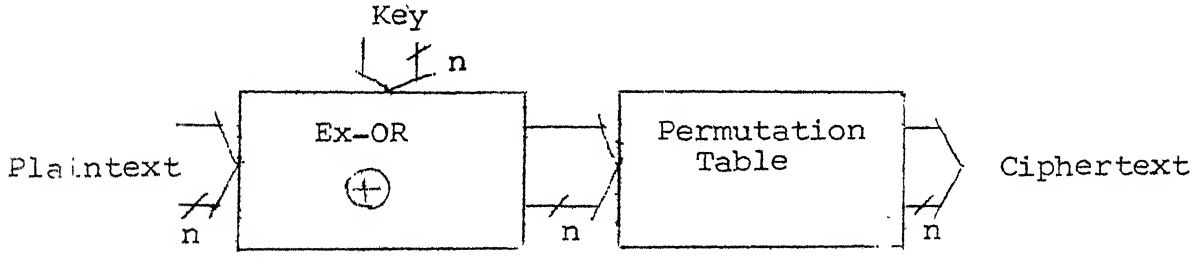


Fig. 5.3 Dyadic Shift System

Both addition mod N and Ex-OR networks require hardware which is linear with the size of the binary words that are mixed. However permutation table still requires a storage of $nx2^n$ bits.

However if these good permutation tables could be obtained algorithmically using a hardware of polynomial complexity the

we can use them for the block cipher. Over and above this, if iterations of these basic structure generate the whole group or a major portion of S_N then this provides a very good, hardware realisable design of a block cipher.

5.2 Quasi-random Permutations through Binary Sequences of length 2^n

It is possible to construct quasi-random permutation tables using binary sequence of length 2^n.

(eg)

Plaintext		Permuted output	
Binary	Decimal	Binary	Decimal
000	0	000	0
001	1	100	4
010	2	010	2
011	3	001	1
100	4	101	5
101	5	111	7
110	6	110	6
111	7	011	3

Each column of the permuted output (binary) is a 2^n length binary sequence. To qualify for a permutation table the binary sequence columns should posses the following properties.

- There should be equal number of ones and zeros in the sequence
- There should be equal number of matches and mismatches of symbols between any two column constituting the table

Hence a good method to construct quasi-random tables is to construct 2^n length binary sequences meeting the above requirements and which are random like & white. If we construct with sequences having good auto correlation, either dyadic or cyclic, then we can use the corresponding shifts as keys as well.

Binary sequences of length 2^n-1, possessing two level near-ideal auto correlation properties, have been well studied in literature [19]. Pseudo random binary sequences (PRBS) or the maximal length sequences (m-sequences) are well known in communication. In statistics design of experiments widely use symmetric designs with parameters (v,k,λ). Symmetric design with parameters $(2^n-1,\ 2^{n-1},\ 2^{n-2})$ are the same as m-sequences.

5.2.1 Binary Sequence Correlation

Correlation between any two $\{0,1\}$ binary sequences $\{a_n\}$, $\{b_n\}$ of length N is defined as

$$R_{ab}(k) = \left(\begin{array}{c}\text{Total Number of}\\ \text{symbol agreements}\end{array}\right) - \left(\begin{array}{c}\text{Total number of}\\ \text{symbol disagreements}\end{array}\right)$$

where k = 0,1...N-1. The value of k specifies the amount of cyclic/dyadic shift of the original sequence and k=0 implies original sequence. If the $\{0,1\}$ sequence is converted into a ± 1 sequence by the relation $a_n = 1-2a_n$ then the correlation can be formulated as

$$R_{ab}(k) = \sum_{n=1}^{N} a_n\, b_{n\oplus k}$$

When we use arithmetic subtraction modulo N for $n \ominus k$ then we get cyclic correlation i.e., n-k and $\{b_m\}$ and $\{b_{n-k}\}$ are given by

$\{b_n\} = b_o b_1 b_2 \ldots b_n$ and $\{b_{n-k}\} = b_k b_{k+1} \ldots b_{n+k}$. If we use Ex-OR addition of binary representation of n & k i.e., $n \oplus k$ then we get dyadic correlation between the sequences. Dyadic correlation is defined only for sequences of length a power of 2. On similar lines the auto correlation function for various values of shifts k of a sequence a_n can be defined as

$$R_a(k) = \sum_{n=1}^{N} a_n a_{n \ominus k} \text{ where } k = 0,1 \ldots N-1.$$

Let $\{a_n\}$ be a $\{0,1\}$ N length sequence with m ones. Let $\{a_{n-k}\}$ be the same sequence after a shift of k with n-k taken modulo N. Let λ be the number of places in which $\{a_n\}$ and $\{a_{n-k}\}$ have one in common i.e.,

If $\{a_n\} = a_o a_1 \ldots a_i \ldots a_n$

$\{a_{n-k}\} = a_k a_{1+k} \ldots a_{i+k} \ldots a_{n+k}$

then $a_i = a_{i+k} = 1$ for exactly λ values of i, $0 \leq i \leq N-1$

The arrangement of ones and zero of $\{a_n\}$ and $\{a_{n-k}\}$ is as follows

$\{a_n\}$	1...1	1...1	0...0	0...0
$\{a_{n-k}\}$	$\underbrace{1...1}_{\lambda}$	$\underbrace{0...0}_{m-\lambda}$	$\underbrace{1...1}_{m-\lambda}$	$\underbrace{0...0}_{N-\lambda-2(m-\lambda)}$

Then the auto correlation function is given by

$R_a(k)$ = (Total number of places where symbols match) - (Total number of places where symbols mismatch

$= (\lambda+N-\lambda-2(m-\lambda)) - (2(m-\lambda))$

$= N-4m+4\lambda$

For any sequence $\{a_n\}$ of length N, $R_a(0) = N$ (i.e., k=0 the zero shift correlation). The auto correlation function $R_a(k)$ $k \neq 0$ will be a constant, if we could construct sequence for which , the number of places of common ones between $\{a_n\}$ and $\{a_{n-k}\}$, is fixed for all values of k, $k \neq 0$. In such a construction we get a binary sequence of length N with m ones and have two level auto correlation. If $R_a(k)$ for $k \neq 0$ has a value zero or close to zero then we get a good correlation sequence that 'Looks' white. A noise like white sequence will have a very high (k=0) correlation and zero out of phase correlation ($k \neq 0$).

5.2.2 Difference Set Concepts and Construction

The concept of difference set helps us in constructing a good correlation sequences and also in analysing its properties [25, pp. 19-20].

Definition 5.1: A set of m residues $D = (a_1 \ldots a_m)$ modulo N is called a (N,m,λ)-difference set if among the collection of elements $[a_i - a_j : i \neq j, 1 \leq i, j \leq m]$ all the non-zero residues occur λ times.

Definition 5.2: The incidence matrix $A = (e_{ij})$ of a (N,m,λ)-difference set is defined as a NxN matrix with elements

$$a_{ij} = \begin{cases} 1 \text{ if } a_j - a_i \in D \\ 0 \text{ if } a_j - a_i \notin D \end{cases}$$

Lemma 5.1: In a (N,m,λ)-difference set the $m(m-1)$ differences get partitioned into $(N-1)$ groups, each having λ differences.

so $(N-1)\lambda = m(m-1)$

Example :

$(7,4,2)$ difference set

$D = (2,4,5,6)$

1 - (5-4) (6-5)

2- (4-2) (6-4)

3 - (5-2) (2-6)

4 - (6-2) (2-5)

5 - (2-4) (4-6)

6 - (4-5) (5-6)

Hence each non zero member is expressed as difference in exactly two ways.

Incidence matrix $A = (a_{ij})$

$A =$

i \ j	0	1	2	3	4	5	6
0	0	0	1	0	1	1	1
1	1	0	0	1	0	1	1
2	1	1	0	0	1	0	1
3	1	1	1	0	0	1	0
4	0	1	1	1	0	0	1
5	1	0	1	1	1	0	0
6	0	1	0	1	1	1	0

Each row of the incidence matrix is a cyclic shifted version of another row. Between any two of the rows the number of ones coincide only in 2 (λ) places and hence the (7,4,2) construction yields a binary sequence $\{a_n\}$ with two level auto correlation function $R_a(0) = 7$ and $R_a(k) = -1$ for $k \neq 0$.

It is possible to construct (N,m,λ) - difference set with $N = 2^n-1$, $m = 2^{n-1}$, $\lambda = 2^{n-2}$ for any value of n, as Lemma 5.1 is satisfied i.e., $(N-1)\lambda = m(m-1)$

$$(2^n-2).2^{n-2} = 2^{n-1}(2^{n-1} - 1)$$

If such a difference is constructed then we can obtain a $\{0,1\}$ sequence of length 2^n-1, having 2^{n-1} ones at locations specified by the difference set D and each non zero member $(1...(2^n-2))$ can be expressed exactly in 2^{n-2} ways. Such a sequence will have a near-ideal correlation property of $R_a(0) = 2^n-1$ for k=0 i.e., zero shift and $R_a(k) = N-4m+4$

$$= 2^n-1 - 4.2^{n-1} + 4.2^{n-2}$$

$$= -1$$

for shifts $k = 1,2,...(2^n-2)$., One particular method of construction, known as Singer difference set construction [15,pp.416-421]. exists for a design of $(2^n-1, 2^{n-1}, 2^{n-2})$. The method generates the difference set via Galois Field $GF(2^n)$. Galois field $GF(2^n)$ can be constructed based on primitive polynomial of degree n. The 2^n elements of the field can be expressed as a polynomial of degree $\leq$ n-1 in α, the root of the primitive polymomial of

degree n over GF(2),. Hence all these elements could be represented by n binary digits, the digits representing the coefficients of the polynomial representation of the field elements. Each column of these binary digits, leaving the all zero element, give the difference set. The column of {0,1} gives the (2^n-1) length sequence and the positions of ones in the sequence give the difference set. See Table 5.1 for example.

5.2.3 Properties of m-sequences

The binary sequences of length (2^n-1) constructed by the above method are called maximal length Pseudo random binary sequences (m-sequences) and they possess the following randomness properties. [14,p.336]

Balance property :

In every sequence the number of ones does not differ from the number of zeros by more than 1.

Run length property :

For every sequence, half the runs (runs of all ones or all zeros) have length 1, one fourth have length 2, one-eight have length 3 etc. as long as the number of runs equals or exceeds 1.

Correlation Property :

The correlation $R_a(k)$ is binary valued

$$R_a(k) = \begin{cases} 2^n-1 & k = 0 \\ -1 & k \neq 0 \end{cases}$$

Table 5.1 (15,8,4)-Cyclic Difference Set Generated by $GF(2^4)$. Irreducible polynomial $p(X) = 1+X+X^4$

Power Representation	Polynomial Representation	4 tuple Representation
0	0	0 0 0 0
α^0	1	1 0 0 0
α^1	α	0 1 0 0
α^2	α^2	0 0 1 0
α^3	α^3	0 0 0 1
α^4	$1+\alpha$	1 1 0 0
α^5	$\alpha+\alpha^2$	0 1 1 0
α^6	$\alpha^2+\alpha^3$	0 0 1 1
α^7	$1+\alpha+\alpha^3$	1 1 0 1
α^8	$1+\alpha+\alpha^2$	1 0 1 0
α^9	$\alpha+\alpha^3$	0 1 0 1
α^{10}	$1+\alpha+\alpha^2$	1 1 1 0
α^{11}	$\alpha+\alpha^2+\alpha^3$	0 1 1 1
α^{12}	$1+\alpha+\alpha^2+\alpha^3$	1 1 1 1
α^{13}	$1+\alpha^2+\alpha^3$	1 0 1 1
α^{14}	$1+\alpha^3$	1 0 0 1

D = (0,4,7,8,10,12,13,14) mod 15

D = (1,4,5,7,9,10,11,12) "

D = (2,5,6,8,10,11,12,13) "

D = (3,6,7,9,11,12,13,14) "

Table 5.2 Partition of Differences

Difference set D = (3,6,7,9,11,12,13,14) mod 15

Non zero members (k)	Differences	Correlation $R_a(k)$
1	(7-6)(12-11)(13-12)(14-13)	-1
2	(9-7)(11-9)(13-11)(14-12)	-1
3	(6-3)(9-6)(12-9)(14-11)	-1
4	(7-3)(11-7)(13-9)(3-14)	-1
5	(11-6)(12-7)(14-9)(3-13)	-1
6	(9-3)(12-6)3-12)(13-7)	-1
7	(13-6)(14-7)(6-14)(3-11)	-1
8	(6-13)(7-14)(14-6)(11-3)	-1
9	(3-9)(6-12)(12-3)(7-13)	-1
10	(6-11)(7-12)(9-14)(13-3)	-1
11	(3-7)(7-11)(9-13)(14-3)	-1
12	(3-6)(6-9)(9-12)(11-14)	-1
13	(7-9)(9-11)(11-13)(12-14)	-1
14	(6-7)(11-12)'12-13)(13-14)	-1

This property makes the sequence look 'white' as we would expect this from a truly random sequence.

Partial correlation property : Every sequence possesses the following desirable partial correlation properties [14,pp.547-8].

Theorem 5.1 : Let $\{a\}$ be a maximal length sequence of length N.

Then $\overline{R_a^{W}(k)} = R_a(k) \qquad W \leq N$

where the normalised partial correlation of the sequence is given by

$R_a^{W}(k) = [\frac{1}{W} \sum_{i=0}^{W-1} a(i)a(i-k)]$ and this averaging denoted by overbar, is over all N starting positions

Proof : By definition

$$\overline{R_a^{W}(k)} = \frac{1}{N} \sum_{j=0}^{N-1} [\frac{1}{W} \sum_{i=0}^{W-1} a(i-j)\ a\ (i-j-k)\ ]$$

Interchanging order of summations, we obtain

$$\overline{R_a^{W}(k)} = \frac{1}{W} \sum_{i=0}^{W-1} [\frac{1}{N} \sum_{j=0}^{N-1} a(i-j)\ a\ (i-j-k)\ ]$$

$$= \frac{1}{W} \sum_{i-0}^{W-1} R_a(k) = R_a(k)$$

Hence, the average partial correlation, averaged over all starting positions, is just the periodic correlation function.

Theorem 5.2: Let $\{a\}$ be a maximal length sequence of length N

Then for $k \neq 0$ and $W < N$

$$\mathrm{Var}[\ R_a^{W}(k)\ ] = \frac{1}{W}\ (1 + \frac{1}{N})\ (1 - \frac{W}{N})$$

where the averaging is taken over all N starting positions.

Proof : Now $\mathrm{Var}[\, R_a^{W}(k)\,] = \overline{[\,R_a^{W}(k)\,]^2} - \overline{(\,R_a^{W}(k))}^2$

$$\overline{[\,R_a^{W}(k)\,]^2} = \frac{1}{N}\sum_{j=0}^{N-1}\left[\frac{1}{W}\sum_{i=0}^{W-1} a(i-j)a(i-j-k)\right]^2$$

so that

$$\overline{[\,R_a^{W}(k)\,]^2} = \frac{1}{NW^2}\sum_{j=0}^{N-1}\sum_{i=0}^{W-1}\sum_{R=0}^{W-1} a(i-j)a(i-j-k)a(h-j)a(h-j-k$$

Using shift and add property of maximal length sequence, i.e., any shift of m-sequence is again a m-sequence and addition of two different m-sequences is again a m-sequence, we have

$$\overline{[\,R_a^{W}(k)\,]^2} = \frac{1}{NW^2}\sum_{J=0}^{N-1}\left[\sum_{i=0}^{W-1}\sum_{h=0}^{W-1} a_s(i-j-k)a_s(h-j-k)\right]$$

where a_s is a shifted version of the a sequence. Now break the double sum into a diagonal and an off diagonal expansion

$$\overline{[\,R_a^{W}(k)\,]^2} = \frac{1}{NW^2}\sum_{j=0}^{N-1}\left[\sum_{i=0}^{W-1} 1 + \sum_{\substack{i=0\\ i\neq h}}^{W-1}\sum_{h=0}^{W-1} a_s(i-j-k)a_s(h-j-k)\right]$$

$$= \frac{1}{NW^2}\sum_{j=0}^{N-1}\sum_{i=0}^{W-1} 1 + \frac{1}{NW^2}\sum_{\substack{I=0\\ i\neq h}}^{W-1}\sum_{h=0}^{W-1}\sum_{j=0}^{N-1} a_s(i-j-k)a_s(h-j-k$$

$$= \frac{1}{NW^2}\left[\,NW + (-1)W.(W-1)\right]$$

so that

$$\mathrm{Var}[\, R_a^{W}(k)\,] = \frac{1}{NW^2}\; NW + (-1)W(W-1) \;-\; \left(-\frac{1}{N}\right)^2 \qquad k\neq 0$$

$$= \frac{1}{W}\left(1 + \frac{1}{N}\right)\left(1 - \frac{W}{N}\right)$$

By Chebyshev inequality [1,p.931] probability distribution functions, the value of $R_a^W(k)$ lying between $\overline{R_a^W(k)} \pm C\sigma$, where C is a constant and σ the standard deviation ($\sigma = \sqrt{\text{Var}[R_a^W(k)]}$) is given by $F(\overline{R_a^W(k)}+C\sigma) - F(\overline{R_a^W(k)}-C\sigma) \geq 1 - \frac{1}{C^2}$, where F(x) denotes the cumulative distribution function c.d.f. of the random variable X. Hence $R_a^W(k)$ has 0.75 probability to lie within $\overline{R_a^W(k)} \pm 2\sigma$, 0.8899 probability to lie within $R_a^W(k) \pm 3\sigma$, 0.9375 probability to lie within $\overline{R_a^W(k)} \pm 4\sigma$ and so on. Chebyshev inequality provides lower bounds on $R_a^W(k)$. See table 5.3 for the case of a 15-length sequence.

5.2.4 Near-ideal 2^n Length Sequences

For constructing a good permutation table which has good cyclic correlation for all its non zero shifts, we are to look for 2^n length sequences with desired properties. We could build the table in case we have $(2^n, 2^{n-1}, 2^{n-2})$ -cyclic difference sets. However difference sets with these parameters $N=2^n$, $m=2^{n-1}$ & $\lambda = 2^{n-2}$ are nonexistant. The parameters do not satisfy Lemma 5.1 i.e. $(N-1)\lambda = m(m-1)$. Put in otherwords the number of differences $a_i - a_j$ $i \neq j$, $1 \leq i,j \leq m$ of the set $D = (a_1, a_2 \ldots a_m)$ mod N can not be divided equally among the 2^n-1 non zero values.

Total number of differences $= m(m-1)$

$= 2^{n-1}(2^{n-1}-1)$

Total nonzero elements $= N-1$

$= 2^n-1$

and $2^{n-1}(2^{n-1}-1) \not\mid 2^n-1$

Table 5.3 Partial Correlation Results of 15 length Sequence.

$N = 15 \qquad \overline{R_a^W(k)} = -\frac{1}{15} = -0.066$

Partial Correlation width W	$Var[R_a^W(k)]$	Standard Deviation $\sigma = \sqrt{Var[R_a^W(k)]}$
15	0	0
14	0.005	0.0712
13	0.0109	0.1046
12	0.017	0.1333
11	0.025	0.1608
10	0.035	0.1885
9	0.047	0.2177
8	0.062	0.2494
7	0.081	0.285
6	0.106	0.3265
5	0.14	0.3771
4	0.19	0.4422
3	0.28	0.5333
2	0.46	0.6798
1	0.99	0.9977

i.e. the total number of differences is not divisible by number of non zero elements. Hence it is not possible to contruct 2^n length $\{0,1\}$ sequences which has near-ideal two level correlation properties $R_a(k) = 2^n$ for k=0 and $R_a(k) = 0$ for $k \neq 0$. However it is possible to partition the differences among the (2^n-1) non zero elements in such a way that we can get near-ideal 3 level correlation properties. If 2^{n-2} nonzero elements can be expressed by $(2^{n-2}-1)$ differences each and the rest of $(2^n-1)-2^{n-2}$ non zero elements by 2^{n-2} differences each then such a 3 level correlation sequences is possible.

Number of differences for 2^{n-2} elements = $(2^{n-2}-1).2^{n-2}$

Number of differences for the rest of elements

$$= 2^{n-2}.\ (2^n-1)-2^{n-2}$$

Summation of the differences $= 2^{n-2}\ [(2^{n-2}-1)+(2^n-1)-2^{n-2}]$

$$= 2^{n-2}\ (2^n-2)$$

$$= 2^{n-1}\ (2^{n-1}-1)$$

$$= \text{Total number of differences.}$$

The sequence thus constructed will have the correlation.

$$R_a(k) = 2^n \qquad k = 0$$

$$R_a(k) = N - 4m + 4\lambda$$

$$= N - 4.\ \frac{N}{2} + 4\lambda$$

$$= -N + 4\lambda$$

$$= \begin{cases} -N + 4.2^{n-2} \text{ i.e. } 0 & \text{for } k \neq 0 \text{ and for } (2^n-1).2^{n-2} \text{ values of } k \\ -N + 4.(2^{n-2}-1) \text{ i.e. } -4 & \text{for } k \neq 0 \text{ and for } 2^{n-2} \text{ values of } k. \end{cases}$$

See Table 5.4 for illustration of 2^4 sequence. In general if we could construct sequences, satisfying the differences partition as mentioned for any value of n, then we can construct the permutation table. The sequence and the shifts of the sequence which give $R_a(k) = 0 \quad k \neq 0$ could be used as columns of the permutation table.

However a general solution to $[2^n, 2^{n-1}, (2^{n-2}, 2^{n-2}-1)]$ -cyclic difference sets has not been attempted so far. Even if we could solve instances of this problem, they will only yield to a table look up implementation requiring many 2^nxn bits storage, in the absence of a systematic algorithmic solution yielding a hardware of reduced complexity.

5.2.5 Acceptable 2^n Length Sequences

We have seen that, based on primitive polynomials of degree n, we can construct 2^n-1 length binary sequences possessing near-ideal correlation property $R_a(k) = \begin{cases} 2^n-1 & k = 0 \\ -1 & k \neq 0 \end{cases}$. These designs partition the $(2^{n-1})(2^{n-1}-1)$ differences into 2^n-2 sets of 2^{n-2} difference each, thus each non zero element is expressed in exactly 2^{n-2} ways. These designs contain 2^{n-1} ones and $2^{n-1}-1$ zeroes and the corresponding $(2^n-1, 2^{n-1}, 2^{n-2})$-

cyclic difference set $D = (a_1 a_2 \ldots a_m) \bmod (2^n-1)$; $m = 2^{n-1}$ contains 2^{n-1} members. We could retain the same difference set $D = (a_1 a_2 \ldots a_m)$ and could try to construct 2^n length sequence by adding one more zero thus making 2^{n-1} ones & 2^{n-1} zeros. However now the differences are to be expressed $(a_i - a_j) \bmod 2^n$. If this method yields ideal partition of differences then we can realise near-ideal 2^n length sequences through the $(2^n, 2^{n-1}, (2^{n-2}, 2^{n-2}-1))$-cyclic difference set thus obtained. This near-ideal 2^n length sequences possess three level correlation property

$$R_a(k) = \begin{cases} 2^n & k = 0 \\ 0 & k \neq 0; \text{ at } ((2^n-1) - 2^{n-2}) \text{ places} \\ -4 & k \neq 0; \text{ at } 2^{n-2} \text{ places.} \end{cases}$$

On the otherhand we can accept the 2^n lengthdesign extended from (2^n-1) length design if the correlations for the non zero shifts are within certain tolerance with respect to the one theoretically obtainable. This will yield a design of $(2^n, 2^{n-1}, ((2^{n-2}, 2^{n-2}-1) \pm \Delta))$ cyclic difference set. The tolerance could be say 1 or 2. When the tolerance is $\Delta = 1$ we get $(2^n, 2^{n-1}, (2^{n-2}, 2^{n-2}-1) \pm 1)$ - cyclic difference set and the 2^n length sequence constructed will have the correlation property of

$$R_a(k) = \begin{cases} 2^n & k = 0 \\ 0, -4, 4, -8 & k \neq 0 \end{cases}$$

We could treat these 2^n length sequences as 'acceptable sequences' in an asymptotic sense i.e. as n becomes large the correlation values for non zero shifts are close to 0 and for zero shift is 2^n.

Table 5.4 Partition of Differences and Correlation of 2^4 Length Sequence
Primitive Polynomial $p(X) = 1+X+X^4$

D = (3,6,7,9,11,12,13,14) mod 16

Non zero member k	Differences	Correlation $R_a(k)$
1	(7-6(12-11)(13-12)(14-13)	0
2	(9-7)(11-9)(13-11)(14-12)	0
3	(6-3)(9-6)(12-9)(14-11)	0
4	(7-3)(11-7)(13-9)	-4
5	(11-6((12-7)(14-9)(3-14)	0
6	(9-3)(12-6)(13-7)(3-13)	0
7	(13-6)(14-7)(3-12)	-4
8	(14-6)(11-3)(3-11)((-14)	0
9	(12-3)(6-13)(7-14)	-4
10	(13-3)(3-9)(6-12)(7-13)	0
11	(14-3)(6-11)(7-12)(9-14)	0
12	(3-7)(7-11)(9-13)	-4
13	(3-6)(6-9)(9-12)(11-14)	0
14	(7-9)(9-11)(11-13)(12-14)	0
15	(6-7)(11-12)(12-13)(13-14)	0

2^{4-2} = 4 non zero members are expressed as differences of 3 each i.e. the members 4,7,9,12.

$(2^4-1)-2^2$ = 11 non zero members are expressed as differences of 4 each, i.e. the members 1,2,3,5,6,8,10,11,13,14,15.

Permutation table could be formed by using any 4 of the sequence of shifts k = 0,1,2,3,5,6,8,10,11,13,14,15

We will try to extend the (2^n-1) length designs obtained through $GF(2^n)$ and see whether the extended 2^n length sequences are 'acceptable sequences'.

Let F denote the 'forward difference' and let B denote the 'back difference' between two elements of the difference set

$$a_i - a_j = F \quad \text{if} \quad a_i > a_j$$
$$= B \quad \text{if} \quad a_i < a_j$$

With this definition of the differences each non zero element gets represented by some mix of F and B. Further for two elements a_i and a_j if $(a_i-a_j) = k$ then $(a_j-a_i) = N-k$, all differences taken modulo N. Hence the whole table of partition of differences becomes symmetrical about the middle. If an element k is represented by a set of Fs & Bs then N-k gets represented by the same set of F's and B's with reversal i.e. $F \Longleftrightarrow B$. See Table 5.5. When the same difference set is retained i.e. $D = (a_1, a_2 \ldots a_m)$; $m = 2^{n-1}$ and the (2^n-1) length is extended to 2^n length design then the differences partition among the non zero elements follow the rule : The 'forward differences' Fs get retained at the same locations and 'Back differences' Bs get shifted one place down in the list of non zero elements. i.e.

if $a_i > a_j$ $\quad (a_i-a_j) \bmod (2^n-1) = (a_i-a_j) \bmod 2^n = F_h$

where F_h indicates 'forward difference' of value h

and if $a_i < a_j$ $\quad (a_i-a_j) \bmod (2^n-1) = B_h$

Table 5.5 (2^4-1)-Design Primitive polynomial $p(X)=1+X+X^4$

$D = (3,6,7,9,11,12,13,14) \bmod (2^4-1)$

Non zero members k	Differences				Total Fs	Bs
1	(7-6) F	(12-11) F	(13-12) F	(14-13) F	4	0
2	(9-7) F	(11-9) F	(13-11) F	(14-12) F	4	0
3	(6-3) F	(9-6) F	(12-9) F	(14-11) F	4	0
4	(7-3) F	(11-7) F	(13-9) F	(3-14) B	3	1
5	(11-6) F	(12-7) F	(14-9) F	(3-13) B	3	1
6	(9-3) F	(12-6) F	(13-7) F	(3-12) B	3	1
7	(13-6) F	(14-7) F	(3-11) B	(6-14) B	2	2
8	(14-6) F	(11-3) F	(6-13) B	(7-14) B	2	2
9	(12-3) F	(3-9) B	(6-12) B	(7-13) B	1	3
10	(13-3) F	(6-11) B	(7-12) B	(9-14) B	1	3
11	(14-3) F	(3-7) B	(7-11) B	(9-13) B	1	3
12	(3-6) B	(6-9) B	(9-12) B	(11-14) B	0	4
13	(7-9) B	(9-11) B	(11-13) B	(12-14) B	0	4
14	(6-7) B	(11-12) B	(12-13) B	(13-14) B	0	4

becomes $(a_i - a_j) \bmod 2^n = B_{h+1}$ for 2^n length design. B_h the Back difference of value h of (2^n-1) length design becomes B_{h+1} back difference of value h+1 of 2^n length design as the modulus changes from (2^n-1) to 2^n. See Table 5.6 for an extended 2^4 design from (2^4-1) design.

The extended design of 2^4 length sequence, derived from retaining the same difference set obtained from (2^4-1) length design, given in Table 5.6, is a near-ideal sequence. This is because the correlation function values of the extended design exactly meets (16,8(4,3)) -cyclic difference set construction. In general a $(2^n-1, 2^{n-1}, 2^{n-2})$-cyclic difference set will lead to near-ideal 2^n length sequence design through $(2^n, 2^{n-1}(2^{n-2}, 2^{n-2}-1))$-cyclic difference set, if the 'F' forward difference' and B'back difference' partition follows the following postulates.

1. The F portion of the partition should be monotinically decreasing in width for increasing values of nonzero elements and should never increase.
2. There should be exactly 2^{n-2} places at which the width of F falls by 1.

The extended 2^n design has correlation function value of $R_a(k) = -1$ at these steps and $R_a(k) = 0$ at all other places. See Fig. 3.1 for a partition leading to near-ideal 2^n length sequence.

The extended design will lead to 'acceptable sequence' of length 2^n if the postulates 1 and 2 cited above are not violated

Table 5.6 2^4-Design extended from (2^4-1)-Design

D = (Primitive polynomial $p(X) = 1+X+X^4$

D = (3,6,7,9,11,12,13,14) Mod (2^4)

Nonzero members k	Differences				Total Fs	Total Bs	Correlations $R_a(k)$
1	(7-6) F	(12-11) F	(13-12) F	(14-13) F	4	0	0
2	(9-7) F	(11-9) F	(13-11) F	(14-12) F	4	4	0
3	(6-3) F	(9-6) F	(12-9) F	(14-11) F	4	0	0
4	(7-3) F	(11-7) F	(13-9) F		3	0	-4
5	(11-6) F	(12-7) F	(14-9) F	(3-14) B	3	1	0
6	(9-3) F	(12-6) F	(13-7) F	(3-13) B	3	1	0
7	(13-6) F	(14-7) F		(3-12) B	2	1	-4
8	(14-6) F	(11-3) F	(3-11) B	(6-14) B	2	2	0
9	(12-3) F		(6-13) B	(7-14) B	1	2	-4
10	(13-3) F	(3-9) B	(6-12) B	(7-13) B	1	3	0
11	(14-3) F	(6-11) B	(7-12) B	(9-14) B	1	3	0
12		(3-7) B	(7-11) B	(9-13) B	0	3	-4
13	(3-6) B	(6-9) B	(9-12) B	(11-14) B	0	0	0
14	(7-9) B	(9-11) B	(11-13) B	(12-14) B	0	4	0
15	(6-7) B	(11-12) B	(12-13) B	(13-14) B	0	4	0

by more than the tolerence Δ. The postulates can be modified to read as follows.

1. The F partition should have a general trend of decreasing width with increasing value of non zero elements.

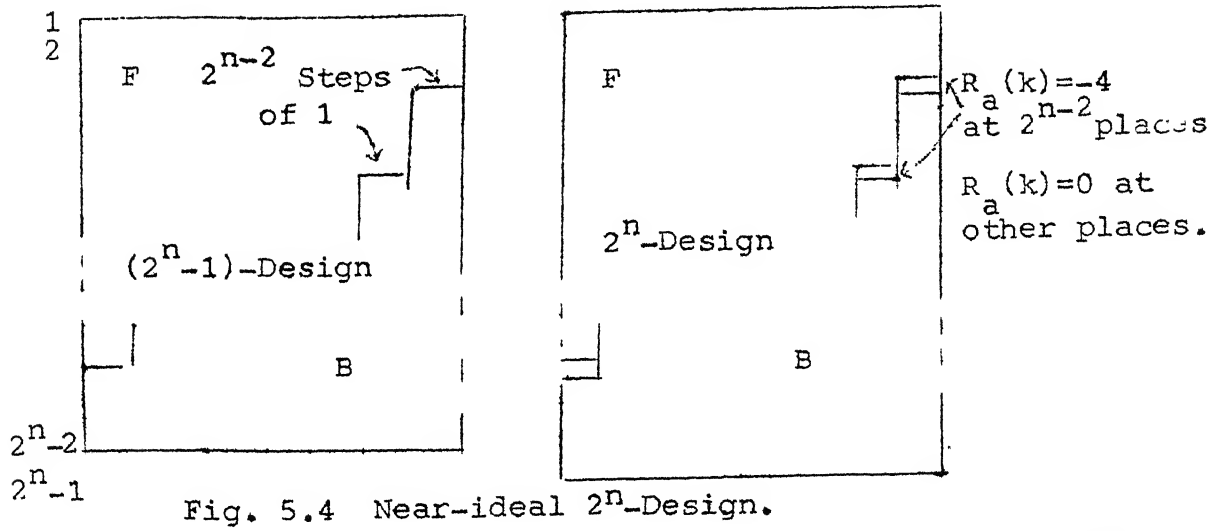


Fig. 5.4 Near-ideal 2^n-Design.

2. At no place the steps in F width can increase by more than tolerance Δ and decrease by more than $(1+\Delta)$. There is no restriction on the number of places of occurrence of step.

The correlation function values will be $R_a(k) = 4$ (for $\Delta = 1$) at places where F width increases by Δ, $R_a(k) = -8$ at places where F decreases by $(1+\Delta)$, $R_a(k) = -4$ at places where F decreases by Δ and $R_a(k) = 0$ for all other non zero shifts. Thus such a partiton of differences leads to $(2^n, 2^{n-1}, (2^{n-2}, 2^{n-2}-1) \pm \Delta)$-cyclic difference set, which leads to 'acceptable sequence' designs of length 2^n. The partitions of differences belonging to Singer construction via $GF(2^n)$ lead to 'acceptable sequence' designs. See Tables 5.7 to 5.11.

Table 5.7 2^5-Design through Primitive Polynomial

$p(X) = 1 + X + X^2 + X^3 + X^5$

$D = (4,6,7,9,11,15,16,17,19,20,21,22,23,26,29,30)$

Nonzero elements k	(2^n-1) length sequence Difference partiton	Fs	Fs	2^n length sequence Bs	Fs	$R_a(k)$
					8	0
1	F F F F F F F F	8	0	0	8	0
2	F F F F F F F F	8	0	0	8	0
3	F F F F F F F F	8	0	0	8	0
4	F F F F F F F F	8	0	0	7	-4
5	F F F F F F F | B	7	1	1	7	0
6	F F F F F F F | B	7	1	1	7	0
7	F F F F F F F | B	7	1	1	6	-4
8	F F F F F F | B B	6	2	2	6	0
9	F F F F F F | B B	6	2	2	7	+4
10	F F F F F F F | B	7	1	1	6	-4
11	F F F F F F | B B	6	2	2	5	-4
12	F F F F F | B B B	5	3	3	6	+4
13	F F F F F F | B B	6	2	2	5	-4
14	F F F F F | B B B	5	3	3	5	0
15	F F F F F | B B B	5	3	3	3	-8
16	F F F | B B B B B	3	5	5	3	0
17	F F F | B B B B B	3	5	5	2	-4
18	F F | B B B B B B	2	6	6	3	+4
19	F F F | B B B B B	3	5	5	2	-4
20	F F | B B B B B B	2	6	6	1	-4
21	F | B B B B B B B	1	7	7	2	+4
22	F F | B B B B B B	2	6	6	2	0
23	F F | B B B B B B	2	6	6	1	-4
24	F | B B B B B B B	1	7	7	1	0
25	F | B B B B B B B	1	7	7	1	0
26	F | B B B B B B B	1	7	7	0	-4
27	B B B B B B B B	0	8	8	0	0
28	B B B B B B B B	0	8	8	0	0
29	B B B B B B B B	0	8	8	0	0
30	B B B B B B B B	0	8			
31						

Table 5.8 2^5-Design through $p(X) = 1 + X + X^3 + X^4 + X^5$

$D = (4,5,7,9,12,16,18,19,20,21,22,24,25,28,29,30)$

Nonzero elements	(2^n-1) length sequence			2^n length sequence		
k	Difference partion	Fs	Bs	Bs	Fs	$R_a(k)$
1	F F F F F F F F	8	0		8	0
2	F F F F F F F F	8	0	0	8	0
3	F F F F F F F F	8	0	0	8	0
4	F F F F F F F F	8	0	0	8	0
5	F F F F F F F | B	7	1	0	7	-4
6	F F F F F F | B B	6	2	1	6	-4
7	F F F F F F | B B	6	2	2	6	0
8	F F F F F F | B B	6	2	2	6	0
9	F F F F F F | F B	7	1	2	7	+4
10	F F F F F | B B B	5	3	1	5	-8
11	F F F F F | B B B	5	3	3	5	0
12	F F F F F | F B B	6	2	3	6	+4
13	F F F F F | B B B	5	3	2	5	-4
14	F F F F | B B B B	4	4	3	4	-4
15	F F F F | B B B B	4	4	4	4	0
16	F F F F | B B B B	4	4	4	4	0
17	F F F F | B B B B	4	4	4	4	0
18	F F F | B B B B B	3	5	4	3	-4
29	F F | B B B B B B	2	6	5	2	-4
20	F F F | B B B B B	3	5	6	3	+4
21	F F F | B B B B B	3	5	6	3	0
22	F | B B B B B B B	1	7	5	1	-8
23	F F | B B B B B B	2	6	7	2	+4
24	F F | B B B B B B	2	6	6	2	0
25	F F | B B B B B B	2	6	6	2	0
26	F | B B B B B B B	1	7	6	1	-4
27	B B B B B B B B	0	8	7	0	-4
28	B B B B B B B B	0	8	8	0	0
29	B B B B B B B B	0	8	8	0	0
30	B B B B B B B B	0	8	8	0	0
31				8		0

Table 5.9. 2^4-Design through $p(X) = 1 + X^3 + X^5$

D = (4,6,8,9,10,12,13,17,18,19,20,21,24,25,27,30)

Nonzero elements	(2^n-1) length sequence			2^n length sequence		
k	Difference partition	Fs	Bs	Bs	Fs	$R_a(k)$
1	F F F F F F F F	8	0		8	0
2	F F F F F F F F	8	0	0	8	0
3	F F F F F F F F	8	0	0	8	0
4	F F F F F F F F	8	0	0	8	0
5	F F F F F F F B	7	1	0	7	-4
6	F F F F F F F F	8	0	1	8	+4
7	F F F F F F F B	7	1	0	7	-4
8	F F F F F F F B	7	1	1	7	0
9	F F F F F F F B	7	1	1	7	0
10	F F F F F B B B	5	3	1	5	-8
11	F F F F F F B B	6	2	3	6	+4
12	F F F F F F B B	6	2	2	6	0
13	F F F F F B B B	5	3	2	5	-4
14	F F F F B B B B	4	4	3	4	-4
15	F F F F F B B B	5	3	4	5	+4
16	F F F B B B B B	3	5	3	3	-8
17	F F F F B B B B	4	4	5	4	+4
18	F F F B B B B B	3	5	4	3	-4
19	F F B B B B B B	2	6	5	2	-4
20	F F B B B B B B	2	6	6	2	0
21	F F F B B B B B	3	5	6	3	+4
22	F B B B B B B B	1	7	5	1	-8
23	F B B B B B B B	1	7	7	1	0
24	F B B B B B B B	1	7	7	1	0
25	B B B B B B B B	0	8	7	0	-4
26	F B B B B B B B	1	7	8	1	+4
27	B B B B B B B B	0	8	7	0	-4
28	B B B B B B B B	0	8	8	0	0
29	B B B B B B B B	0	8	8	0	0
30	B B B B B B B B	0	8	8	0	0
31				8		0

Table 5.10 2^5 Design through $p(X) = 1 + X^3 + X^5$

D = (1,6,8,10,11,12,14,15,19,20,21,22,23,26,27,29)

Nonzero elements	(2^n-1) length sequence			2^n length sequence		
k	Difference partion	Fs	Bs	Bs	Fs	$R_a(k)$
1	F F F F F F F F	8	0		8	0
2	F F F F F F F F	8	0	0	8	0
3	F F F F F F F B	7	1	0	7	-4
4	F F F F F F F F	8	0	1	8	+4
5	F F F F F F F B	7	1	0	7	-4
6	F F F F F F F B	7	1	1	7	0
7	F F F F F F F F	8	0	1	8	+4
8	F F F F F F F B	7	1	0	7	-4
9	F F F F F F F B	7	1	1	7	0
10	F F F F F B B B	5	3	1	5	-8
11	F F F F F F B B	6	2	3	6	+4
12	F F F F F B B B	5	3	2	5	-4
13	F F F F F B B B	5	3	3	5	0
14	F F F F F B B B	5	3	3	5	0
15	F F F F F B B B	5	3	3	5	0
16	F F F B B B B B	3	5	5	3	0
17	F F F B B B B B	3	5	5	3	0
18	F F F B B B B B	3	5	5	3	0
19	F F F B B B B B	3	5	5	3	0
20	F F B B B B B B	2	6	5	2	-4
21	F F F B B B B B	3	5	6	3	+4
22	F B B B B B B B	1	7	5	1	-8
23	F B B B B B B B	1	7	7	1	0
24	B B B B B B B B	0	8	7	0	-4
25	F B B B B B B B	1	7	8	1	+4
26	F B B B B B B B	1	7	7	1	0
27	B B B B B B B B	0	8	7	0	-4
28	F B B B B B B B	1	7	8	1	+4
29	B B B B B B B B	0	8	7	0	-4
30	B B B B B B B B	0	8	8	0	0
31				8		0

38,42,44,45,46,47,50,52,56,57,62)

Nonzero elements	(2^n-1) length sequence			2^n length sequence	
K	Difference Partition	Fs	Bs	Bs	Fs	$R_s(k)$
1	F F F F F F F F F F F F F F F	16	0		16	0
2	F F F F F F F F F F F F F F F	16	0	0	16	0
3	F F F F F F F F F F F F F F F	16	0	0	16	0
4	F F F F F F F F F F F F F F F	16	0	0	16	0
5	F F F F F F F F F F F F F F F	16	0	0	16	0
6	F F F F F F F F F F F F F F B	15	1	0	15	-4
7	F F F F F F F F F F F F F F B	15	1	1	15	0
8	F F F F F F F F F F F F F F B	15	1	1	15	0
9	F F F F F F F F F F F F F F B	15	1	1	15	0
10	F F F F F F F F F F F F F F B	15	1	1	15	0
11	F F F F F F F F F F F F F B B	14	2	1	14	-4
12	F F F F F F F F F F F F F B B	14	2	2	14	0
13	F F F F F F F F F F F F B B B	13	3	2	13	-4
14	F F F F F F F F F F F F B B B	14	2	3	14	+4
15	F F F F F F F F F F F F B B B	13	3	2	13	-4
16	F F F F F F F F F F F F B B B	13	3	3	13	0
17	F F F F F F F F F F F F B B B	13	3	3	13	0
18	F F F F F F F F F F F B B B B	12	4	3	12	-4
19	F F F F F F F F F F F F B B B	13	3	4	13	+4
20	F F F F F F F F F F F F B B B	13	3	3	13	0
21	F F F F F F F F F F B B B B B	11	5	3	11	-8
22	F F F F F F F F F F B B B B B	11	5	5	11	0
23	F F F F F F F F B B B B B B B	9	7	5	9	-8
24	F F F F F F F F F B B B B B B	10	6	7	10	+4
25	F F F F F F F F B B B B B B B	9	7	6	9	-4
26	F F F F F F F F B B B B B B B	9	7	7	9	0
27	F F F F F F F B B B B B B B B	8	8	7	8	-4
28	F F F F F F F F B B B B B B B	9	7	8	9	+4
29	F F F F F F F F F B B B B B B	10	6	7	10	+4
30	F F F F F F F F B B B B B B B	9	7	6	9	+4
31	F F F F F F F B B B B B B B B	7	9	7	7	-8
32	F F F F F F F F B B B B B B B	9	7	9	9	+8
33	F F F F F F F B B B B B B B B	7	9	7	7	-8
34	F F F F F F B B B B B B B B B	6	10	9	6	-4
35	F F F F F F F B B B B B B B B	7	9	10	7	+4
36	F F F F F F F B B B B B B B B	8	8	9	8	+4
37	F F F F F F F B B B B B B B B	7	9	8	7	-4
38	F F F F F F F B B B B B B B B	7	9	9	7	0
39	F F F F F F B B B B B B B B B	6	10	9	6	-4
40	F F F F F F F B B B B B B B B	7	9	10	7	+4
41	F F F F F B B B B B B B B B B	5	11	9	5	-8
42	F F F F F B B B B B B B B B B	5	11	11	5	0
43	F F F B B B B B B B B B B B B	3	13	11	3	-8
44	F F F B B B B B B B B B B B B	3	13	13	3	0
45	F F F F B B B B B B B B B B B	4	12	13	4	+4
46	F F F B B B B B B B B B B B B	3	13	12	3	-4
47	F F F B B B B B B B B B B B B	3	13	13	3	0
48	F F F B B B B B B B B B B B B	3	13	13	3	0
49	F F B B B B B B B B B B B B B	2	14	13	2	-4
50	F F F B B B B B B B B B B B B	3	13	14	3	+4
51	F F B B B B B B B B B B B B B	2	14	13	2	-4
52	F F B B B B B B B B B B B B B	2	14	14	2	0
53	F B B B B B B B B B B B B B B	1	15	14	1	-4
54	F B B B B B B B B B B B B B B	1	15	15	1	0
55	F B B B B B B B B B B B B B B	1	15	15	1	0
56	F B B B B B B B B B B B B B B	1	15	15	1	0
57	F B B B B B B B B B B B B B B	1	15	15	1	0
58	B B B B B B B B B B B B B B B	0	16	15	0	-4
59	B B B B B B B B B B B B B B B	0	16	16	0	0
60	B B B B B B B B B B B B B B B	0	16	16	0	0
61	B B B B B B B B B B B B B B B	0	16	16	0	0
62	B B B B B B B B B B B B B B B	0	16	16	0	0
63				16		0

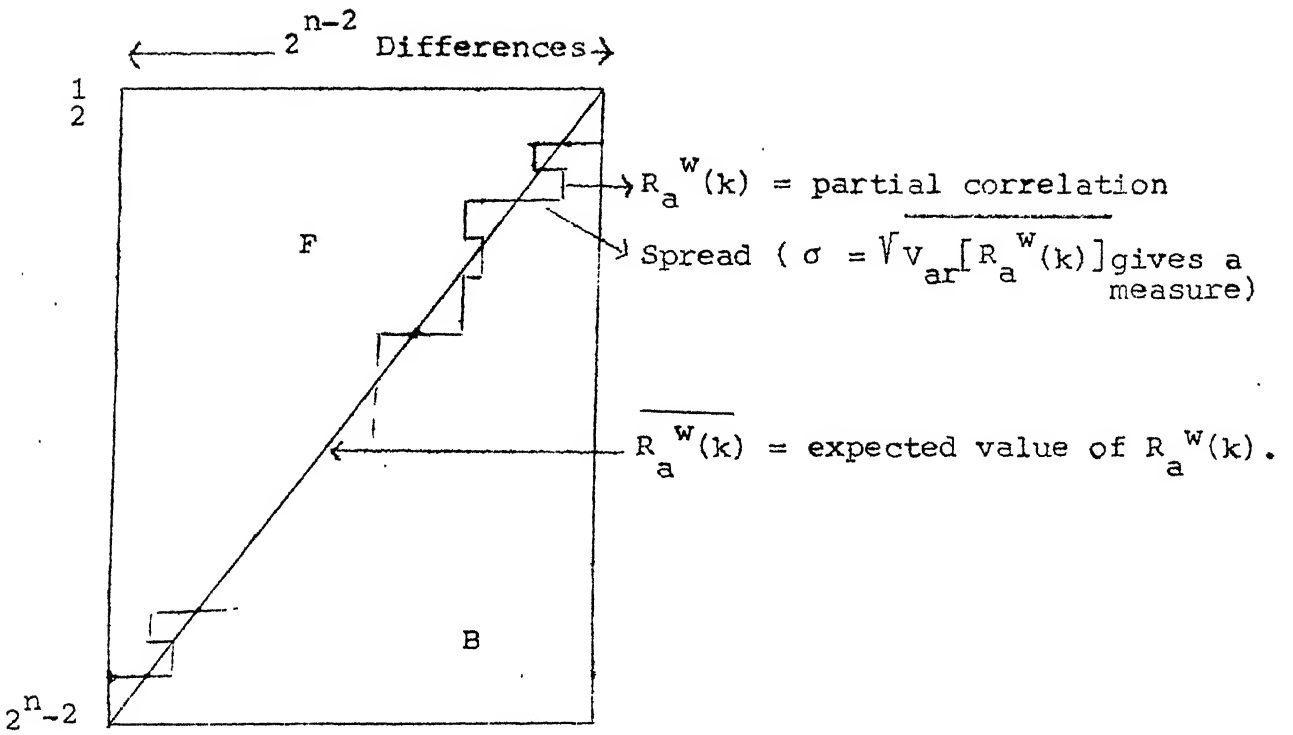


Fig. 5.5 'Acceptable' 2^n-Design

5.2.7 Enumeration of Tables

The construction procedure of field elements of $GF(2^n)$, using primitive polynomials of degree n, gives a good permutation table. Furthermore the 'n' columns of the binary 'n' tuple representation of the 2^n field elements could be permuted among themselves and used as permutation tables. This gives n! tables for every primitive polynomial of degree n.

The number of primitive polynomials of degree n is given by the $\phi\ (2^n-1)/n$ where Euler ϕ-function

$\phi(x)$ is defined by

$$\phi(x) = \begin{cases} 1 & x=1 \\ \prod_{i=1}^{k} p_i^{e_i-1}(p_i-1) & x>1 \ ; \ x = \prod_{i=1}^{k} p_i^{e_i} \\ & \text{(integer x expressed as product of powers of primes)} \\ p-1 & \text{if } x = p \text{ a prime} \end{cases}$$

Exhaustive table of primitive polynomials of degree upto 34 appear in [20,pp.251-270]. Primitive trinomials of degree upto 1000 is given in [28], [29] . See Table 5.12 for the number of permutation tables that can be formed for various values of n.

Table 5.12 Number of Permutation Tables

Degree n	(2^n-1)	Number of primitive polynomials $N_m = \frac{\phi(2^n-1)}{n}$	Number of Permutation tables $N_p = n! \times N_m$
1	1*	1	1
2	3*	1	2
3	7*	2	12
4	15	2	48
5	31*	6	720
6	63	6	4,320
7	127*	18	90,720
8	255	16	645,120
9	511	48	17,418,240
10	1,023	60	2.17728 E + 8
11	2,047	176	7.02535 E + 9
12	1,095	144	6.89762 E + 10
13	8,191*	630	3.02302 E + 12
14	16,383	756	6.59067 E + 13
15	32,768	1,800	2.35381 E + 15
16	65,535	2,048	4.28498 E + 16
17	131,071*	7,710	2.74235 E + 18
18	262,143	8,064	5.16287 E + 19
19	524,287*	27,594	3.35667 E + 21
20	1,048,575	24,000	5.83896 E + 22
21	2,097,151	84,672	4.32597 E + 24
22	4,194,303	120,032	1.34916 E + 26
23	8,388,607	356,960	9.22813 E + 27
24	16,777,215	276,480	1.71541 E + 29

* Mersenne Primes.

CHAPTER 6

HARDWARE IMPLEMENTATION

We have seen in chapter 5, that the n tuple representation of Galois Field GF(2^n) generates a good permutation table. Hence any hardware realisation of the system will involve Galois Field arithmetic. Galois Field arithmetic like addition, multiplication, exponentiation of field elements is essentially carried out by basic operation of binary field arithmetic. This requires basically modulo 2 addition (Ex-OR logical operation) and multiplication (AND logical operation) of binary field elements (0,1). Efficient Galois Field arithmetic hardware architectures are reported in literature. [3],[18],[27],[26] and the trend indicates that polynomial complexity hardware is distinctly possible.

In this chapter we give a hardware scheme to implement the cipher system algorithm discussed in chapter 5. After making an initial approach (Section 6.1) in respect of hardware requirements of Encryption (section 6.1.1) and of decryption (sec. 6.1.2), we alter the basic structure, to circumvent the difficult 'discrete logarithm problem' required in decryption, using involution (section 6.1.3) and we finally arrive at a new structure (section 6.2) Section 6.2.1 presents the Block schematic Subsequent sections 6.2.2 and 6.2.3 explore the set of permuta-

tions generated by the structure and cryptanalysis of the final structure respectively. A hardware, designed using available MSI/LSI chips, and its performance in respect of speed (throughout) and hardware complexity are given in section 6.3.

6.1 Initial Approach

6.1.1 Encryption

We are to use Galois Field GF(2^n) element n- tuple representation and its cyclic shifted versions as the permutation tables, as we have established the usability through (2^n-1, 2^{n-1}, 2^{n-2})-cyclic difference sets. A general sketch of the cipher is given Fig. 6.1.

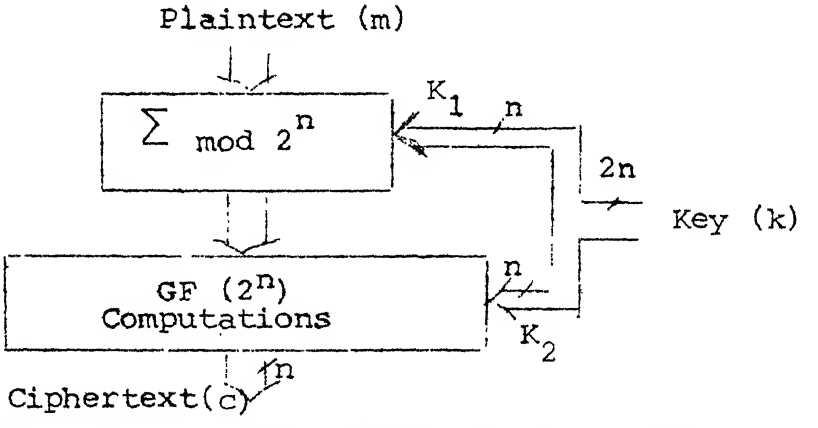


Fig. 6.1 General sketch of the cipher

A n-bit key K_2 specifies the primitive polynomial of degree n plus the column permutation to be used. Galois Field arithmetic operation based on this has to be carried out on n-bit wide arithemetic sum modulo 2^n of n-bit plaintext and n bit key specified by K. The n bit wide output of the GF(2^n) computation

is the cipher text. Here the usable key space is 2^{2n} specified by 2n bit key K. Encryption is carried out by first performing addition modulo 2^n and then Galois Field $GF(2^n)$ computation. Linear complexity (O(n)) hardware exists for carrying out modulo 2^n summation of 2-n bit words. A ripple carry adder/carry look ahead adder could be used for this purpose. Galois Field computation in $GF(2^n)$ involves taking exponentiation of primitive element α (for 2^n-1 elements α^1, $\alpha^2 \ldots \alpha^{2^n-1}$) and also some hardware to deal with 0 element (α^∞) of the field. Exponentiation of any field element in $GF(2^n)$ can be carried out by $GF(2^n)$-multiplier hardware. This hardware can be used to do exponentiation by successive multiplications element α. Thus α^0, α^1, $\alpha^2 \ldots \alpha^{2^n-2}$ can be computed. However this will involve exponential time complexity ($O(2^{n/2})$ on the average. This could be speeded up by using 'left to right binary method of exponentiation [16, p.441]. In this method if one has to compute α^x the following steps are followed. x is first written in binary number system $b_{n-1} \ldots b_1 b_o$, then, after suppressing leading zeros, each '1' is replaced by the pair of letters SX (S = square, X = multiply) and 0 is replaced by the letter S and the 'SX' appearing at the left is deleted. The sequence of operation to compute α^x is given by this string of 'S' and 'X' on the element α, starting from left to right. For example

$$100 = (1100100)_2$$

$$1100100 = SXSXSSSXSS$$

This implies 'square α , multiply byα,square, square, square, multiply by α , square, square

$$\alpha^{100} = (((((\alpha^2\alpha)^2)^2)^2\alpha)^2)^2$$

This can be verified as correct by seeing the power RHS evaluates to ((2+1)2.2.2+1)2.2 = 100. Hence exponentiation can be carried out in linear time complexity O(n) where n represents the number of bits required to represent the power x. The number of multiplications required is bounded by (n-1)+(n-1) = 2(n-1) i.e. (n-1) squares and (n-1) multiply leading to a total of 2(n-1) multiplications. For particular values of x it takes ($\lfloor \log_2 x \rfloor$ -1)+ (γ(x)-1) multiplications, where $\lfloor y \rfloor$ indicates trunclation of y to integer (floor of y) and γ(x) indicates the number of ones in binary representation of x. Exponentiation takes care of generation of α^1, α^2,...α^{2^n-1}. Element $\alpha^\infty = 0$ can be taken care of by appropriate detection circuit and over-riding the GF(2^n) exponentiation. See Fig. 6.2

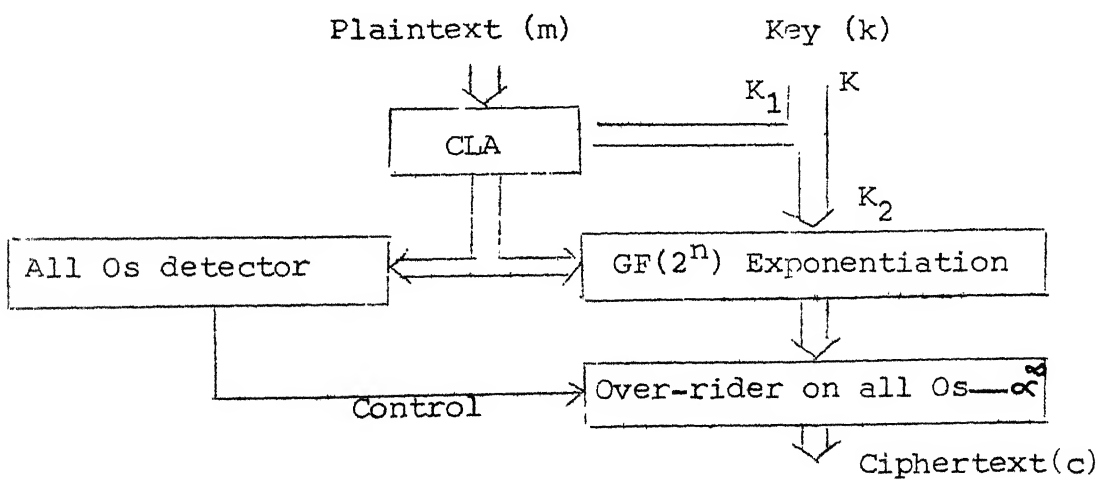


Fig. 6.2 Encryption Hardware Schematic

Hence it appears feasible to realise polynomial time complexity for encryption.

6.1.2 Difficulty in Decryption

Decryption process involves translating the cipher text into plaintext and this can be achieved by essentially tracing the path of encryption in the reverse direction. In the proposed scheme this can be done provided we can carryout inverse operations of $GF(2^n)$ computation and addition modulo 2^n. Inverse operation of addition modulo 2^n can be carried out by a hardware subtractor mod 2^n. In the inverse operation of $GF(2^n)$ computation, the case of element α^∞ could again be taken care by overriding by hardware detector. However the case of elements $\alpha^0, \alpha^1, \alpha^2 \cdots \alpha^{2^n-2}$ pose a harder problem in inverse calculation. In Galois Field $GF(2^n)$, the field elements $A(\alpha) \equiv \alpha^x \bmod p(\alpha)$ take on the value of each non zero field element $A(\alpha)$ in $GF(2^n)$ exactly once, as the value of x ranges through integers $0,1,\ldots 2^n-2$. Conversely, to each nonzero field element $A(\alpha)$ in $GF(2^n)$, we associate the integer x, and say that x is the logarithm of $A(\alpha)$. Since $\alpha^{2^n-1} \equiv 1 \bmod p(\alpha)$, the logarithm is only defined mod 2^n-1. We refer to the calculation of x, $x = \log_\alpha A(\alpha)$ over $GF(2^n)$, as the'discrete logarithm problem' [4,pp 73-82]. One can, using the $GF(2^n)$ multiplier hardware, repeatedly multiply $A(\alpha)$ by α^{-1} and count the number of multiplications required to obtain the element $1=\alpha^0$. This would require exponential time

complexity, on the average about 2^{n-1} time steps. If the field is small enough, one can tabulate all field elements and their logarithms, and use this table for computation within the field, much as one uses a table of natural logarithms for calculation involving real numbers. See Table 6.1 for log table of $GF(2^n)$.

Table 6.1 Log Table for $GF(2^4)$ using $p(X) = 1+X+X^4$

Field element $A(\alpha)$ $\alpha^3\ \alpha^2\ \alpha\ 1$	$\log_\alpha A(\alpha)$
0 0 0 0	α
0 0 0 1	0
0 0 1 0	1
0 0 1 1	4
0 1 0 1	2
0 1 0 1	8
0 1 1 0	5
0 1 1 1	10
1 0 0 0	3
1 0 0 1	14
1 0 1 0	9
1 0 1 1	7
1 1 0 0	6
1 1 0 1	13
1 1 1 0	11
1 1 1 1	12

However for a table-look up scheme we require huge storage of (2^n-1) n-bit words. Few other algorithms, namely Adleman Algorithm [21],[2] Waterloo algorithm [11], & Copper Smith algorithm [45], reported in literature indicate that they all take exponential time complexity to evaluate logarithms in $GF(2^n)$.

6.1.3 Solution through Involution

However the problem of evaluating logarithm can be circumvented in the implementation of the cryptographic system, by modifying the structure to accomodate 'involution' [17, p 236].

Definition : Let n block mean a sequence of 0_s and 1_s of length n $\underline{X} = (X_o, X_1 \ldots X_{n-1}) \in Z_{2,N} \quad Z_{2,N}$ a vector of length N of (0,1). Then a mapping π on $Z_{2,n}$ into $Z_{2,n}$ is called an 'involution' if $\pi^2 = I$, the identity transformation.

Let the transformation f be a mapping on $Z_{2,n}$ then

$\pi_f : (\underline{x},\underline{y}) \longrightarrow (\underline{x} \oplus f(\underline{y}), \underline{y})$.

π_f consists of the following steps

- Transformation of right-half N-block $\underline{y}$ of $(\underline{x},\underline{y})$ by f

  f : $\underline{y} = f(\underline{y})$

- Component by component addition (modulo 2) of this result to the left half N block $\underline{x}$ of $(\underline{x},\underline{y})$ i.e. $f(\underline{y}) \oplus \underline{x}$

- Concatenation of $f(\underline{y}) \oplus \underline{x}$ to the right half of N-block $\underline{y}$ i.e. $(\underline{x} \oplus f(\underline{y}), \underline{y})$

Computing π^2_f we find

$$(\underline{x},\underline{y})\ \pi^2_f = (\underline{x} \oplus f(\underline{y}),\ \underline{y})\ _f$$

$$= (\underline{x} \oplus f(\underline{y}) \oplus f(\underline{y}),\ \underline{y})$$

$$= (\underline{x},\underline{y}) \quad \text{since } f(\underline{y}) \oplus f(\underline{y}) = \underline{0}$$

Thus $_f$ is an involution and is an element of symmetric group S_N. (Proof : If $(\underline{x})\pi = (\underline{y})\pi$ then

$$(\underline{x})\ \pi^2 = \underline{x} = \underline{y} = (\underline{y})\ \pi^2$$

Let γ be intercharge mapping

$$\gamma: \quad (\underline{x},\underline{y}) \longrightarrow (\underline{y},\underline{x})$$

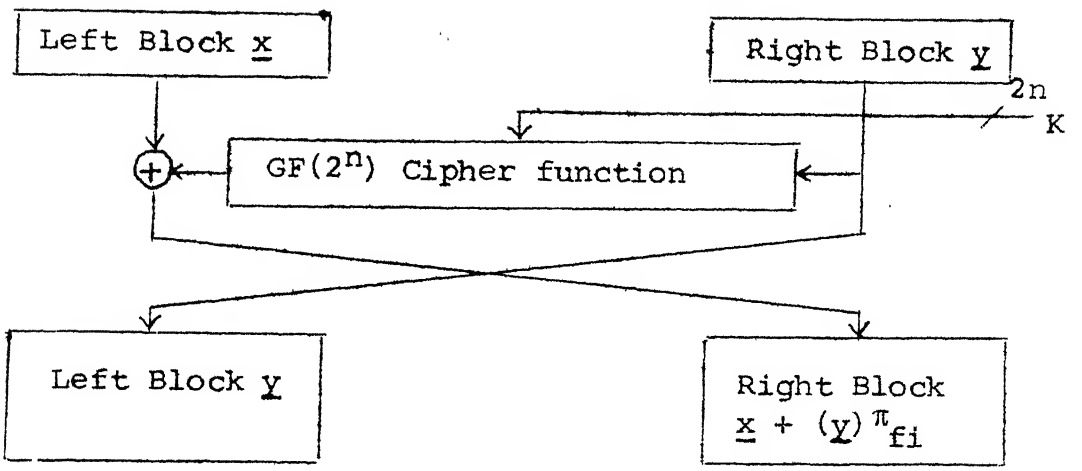
γ is also an 'involution'.

Let a product of these mappings $\pi_p = \pi_{f1}\ \gamma\ \pi_{f2} \cdots \pi_{fm}$. Since $(\pi^2_{fi}) = \gamma^2 = I$ (identity permutation) we have $\pi^{-1}_p = \pi_{fm} \cdots \pi_{f2}\ \gamma\ \pi_{f1}$. Ther encryption is done by π_p and decryption by π^{-1}_p using the same hardware structure. Hence using this 'involution' structure we need only to find exponentiation in $GF(2^n)$, thus avoiding discrete logarithm problem. We simultaneously realise a hardware structure which can be same for encryption and decryption. See fig. 6.3

6.2 Final Structure and its Capabilities

6.2.1 Block Schematic

Using the product of involutions as the basic building block we can finalise the structure. A minimum of 2 rounds will ensure that no part of plaintext appear at the cipher text without going through our $GF(2^n)$ based transformation. Many



One Round of Cipher System :

Fig. 6.3 Product of two Involutions π_{fi} and γ

rounds of this structure could be implemented depending on the speed requirements. See fig. 6.4 for the final block schematic using 4 rounds.

The proposed system requires a 8n bits key in the form of K_1, K_2, K_3 & K_4 each of 2n bits. However by using Keyscheduling algorithm K_1, K_2, K_3 & K_4 could be derived from $\geq$2n bits of key by using shifted versions of key.

6. 2 Permutation Space of the Cipher System

The involution structure we have used in the cipher system contains only 'even' permutations. The mappings used were π_{fi} and γ. We will prove for $n > 1$ the mappings contain

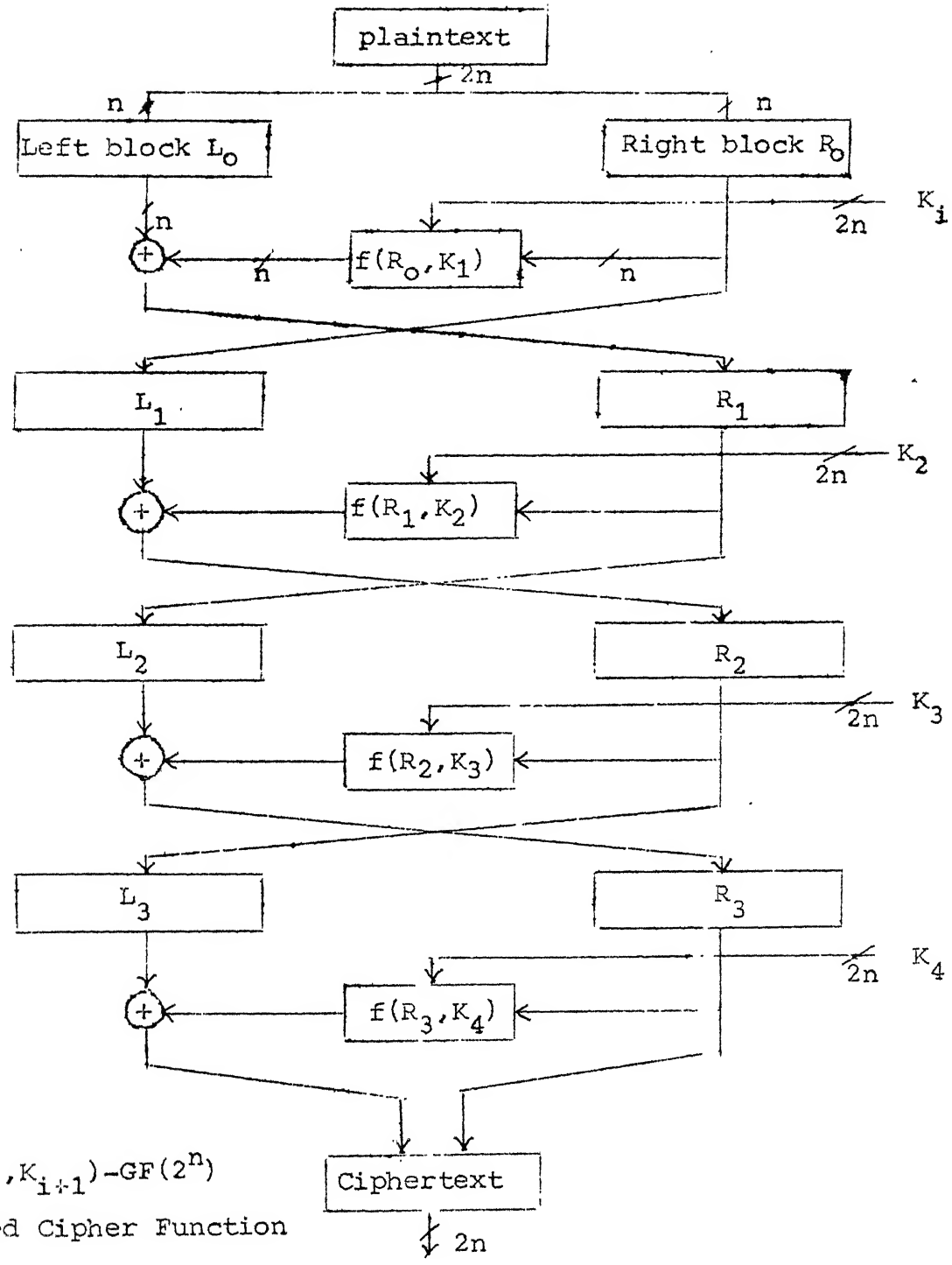
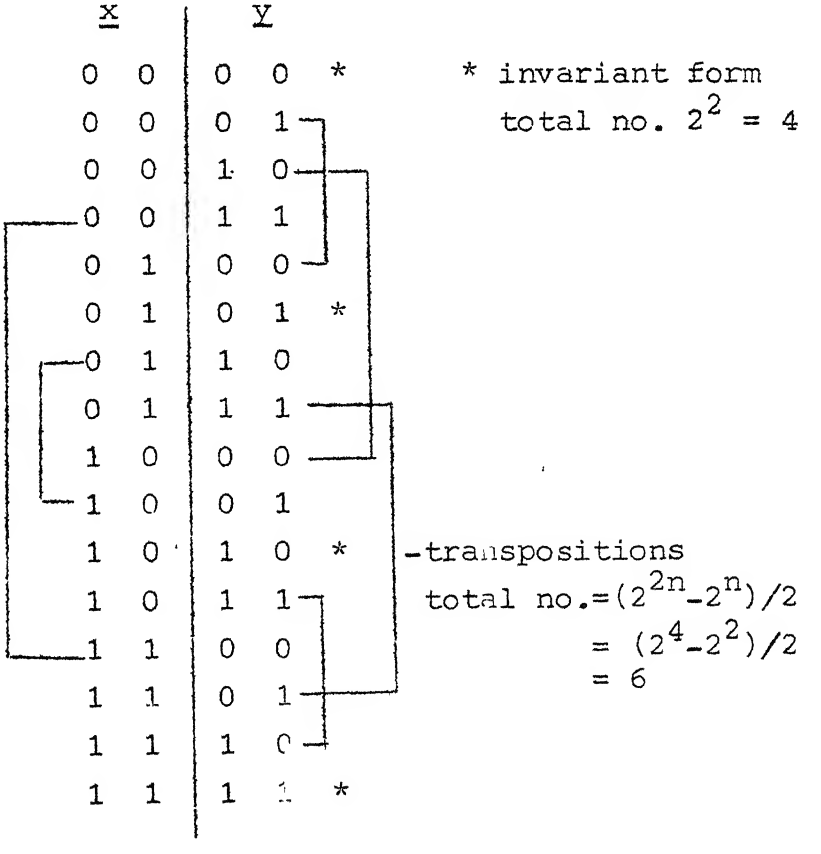


Fig. 6.4 Final Block Schematic of the Cipher System

only even permutations. Lemma 6.1 : For any n > 1, ν is an even permutation. Proof : ν exchanges, in pairs, a vector $(\underline{x},\underline{y})$ and $(\underline{y},\underline{x})$. Hence there is a transposition involved between $(\underline{x},\underline{y})$ and $(\underline{y},\underline{x})$ and no transposition for $(\underline{x},\underline{x})$. $(\underline{x},\underline{x})$ form occurs for 2^n values as both $\underline{x}$, & $\underline{y}$ are n bit binary vector. Hence the no. of transposition is given by $(2^{2n}-2^n)/2$ and is even. 2^{2n} gives the total values, out of which 2^n are invariant under ν, for the remaining $(2^{2n}-2^n)$ the transformation exchanges them in pairs. See table 6.2.

Table 6.2 ν-Even Permutation



Lemma 6.2 : If $n > 1$ then π_{fi} is an even permutation for every f_i.

π_{fi} is a permutation in our case. Hence each vector $\underline{0}$ (all zeros) to $\underline{1}$ (all ones) occur only once in the function $f(R_{i-1}, K_2)$ of the transformation.

Let $\underline{0}$ denote all zero vector then the transformation fixes (invariant) the elements and the rest of the elements are exchanged in pairs. The number of transpositions is given by $(2^{2n}-2^n)/2$ and is even. See table 6.3.

Table 6.3 π_{fi}-Even Permutation

$\underline{x}$	$\underline{y}$	$\underline{x}$	$f(\underline{y})$	$\underline{x} \oplus f(\underline{y})$,	$\underline{y}$	
0 0	0 0	0 0	0 1	0 1	0 0	
0 0	0 1	0 0	0 0	0 0	0 1	*
0 0	1 0	0 0	1 1	1 1	1 0	
0 0	1 1	0 0	1 0	1 0	1 1	
0 1	0 0	0 1	0 1	0 0	0 0	
0 1	0 1	0 1	0 0	0 1	0 1	*
0 1	1 0	0 1	1 1	1 0	1 0	
0 1	1 1	0 1	1 0	1 1	1 1	
1 0	0 0	1 0	0 1	1 1	0 0	
1 0	0 1	1 0	0 0	1 0	0 1	*
1 0	1 0	1 0	1 1	0 1	1 0	
1 0	1 1	1 0	1 0	0 0	1 1	
1 1	0 0	1 1	0 1	1 0	0 0	
1 1	0 1	1 1	0 0	1 1	0 1	*
1 1	1 0	1 1	1 1	0 0	1 0	
1 1	1 1	1 1	1 0	0 1	1 1	

* $2^{n-1} = 2^2 = 4$ places where $f(\underline{y}) = \underline{0}$ and hence invariant

] transpositions, total number $= (2^{2n}-2^2)/2 = 6$.

Hence the cipher system which does several rounds of these ν and π_{fi} transformations lie within the group of even permutations $A_{(2^{2n})}$.

f(y) can be seen as n columns of 2^n length binary sequences and it has been proved in literature [6][10] that if one uses all $2^n.2^n$ functions one can generate $A_2 2n$. In our cipher system we are using only a chosen set of 2^{2n} functions in each round.

6.2.3 Cryptanalysis of the Final Scheme

Ciphertext only attack depends on the side information of the source of plaintext in terms of probable words. It is difficult to maintain a catalogue and collect statistics on 2^n of n bits block as even for n=64 thus would mean $\backsim 1.8 \times 10^{19}$ entries.

The proposed cipher system uses a minimum key space of 2^n keys of length n bits. Hence key trial on 1.8×10^{19} keys is also impractical.

The known plaintext and chosen plaintext attacks correspond to passive and active system identification problems respectively [8] . Unlike in automatic fault diagnosis, the goal here is to build systems which are difficult, rather than easy to identify. The cipher transformation using Galois Field $GF(2^n)$ computations is highly nonlinear with nonlinearity of all orders present (See table 6.4) and hence one requires plaintext and corresponding ciphertext for 2^n elements as the system

transformation is given by $2^n x 2^n$ binary matrix. Then one has to solve the 2^n linearly independent equations to identify the system matrix. Collection of plaintext-ciphertext pairs for 2^n different values itself is infeasible. Further approximately 2^{3n} operations are required to compute the matrix. If the transformation were linear or affine then the system matrix size is only nxn and could be solved in n^3 operations. If n = 64 then this means approximately 0.3 million operation which could be carried out in 0.3 secs. assuming a 1 μs instruction time.

Table 6.4 Reed Muller Canonical Boolean Expression
Uses standard basis for 2^n sequences obtained through $GF(2^5)$.

Sequence : D = (4,6,8,9,10,12,13,17,18,19,20,21,24,25, 27,30)

$$f(x_o x_1 x_2 x_3 x_4) = x_4 \oplus x_4x_o \oplus x_1x_2 \oplus x_3x_1 \oplus x_3x_2 \oplus x_4x_3 \oplus x_4x_1x_o \oplus x_4x_2x_o \oplus x_4x_3x_o \oplus x_3x_2x_1 \oplus x_4x_2x_1 \oplus x_3x_2x_1x_o \oplus x_4x_2x_1x_o \oplus x_4x_3x_2x_o \oplus x_4x_3x_2x_1$$

Sequence : D = (4,6,7,9,11,15,16,17,19,20,21,22,23,26,29,30)

$$f(x_o x_1 x_2 x_3 x_4) = x_4 \oplus x_3x_o \oplus x_4x_o \oplus x_3x_2 \oplus x_4x_2 \oplus x_4x_3 \oplus x_2x_1x_o \oplus x_3x_2x_o \oplus x_4x_3x_o \oplus x_4x_2x_1 \oplus x_4x_3x_1 \oplus x_3x_2x_1x_o \oplus x_4x_2x_1x_o \oplus x_4x_3x_1x_o \oplus x_4x_3x_2x_1$$

Sequence : D = (1,6,8,10,11,12,14,15,19,20,21,22,23,26,27,29)

$$f(x_o x_1 x_2 x_3 x_4) = x_1 \oplus x_2 \oplus x_ox_1 \oplus x_ox_4 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_3x_4 \oplus x_ox_1x_2 \oplus x_ox_1x_3 \oplus x_ox_3x_4 \oplus x_1x_2x_4 \oplus x_2x_3x_4 \oplus x_ox_1x_2x_3 \oplus x_ox_1x_3x_4.$$

6.3 Hardware and its Performance

This section presents a detailed design of the block cipher of length 64. Addition modulo 2^{32} is carried out by 8 4-bit Full Adders connected in series. (Fig. 6.5). To add two 32 bits word, it takes a worst case total of $\sim$ 200 n sec including the time to load the operands using LSTTL ICs. [30], [31]. During the 4 rounds of the cipher this addition has to be carried out as many times.

$GF(2^{32})$ exponentiation is done through repeated multiplications using 'left to right binary method' of taking power of a field element α. $GF(2^{32})$ multiplier uses an array architecture to carryout the multiplication faster. The multiplier first forms the product a.b where $a = (a_{n-1},\ldots,a_o)$ and $b = (b_{n-1},\ldots,b_o)$ are two field elements. A (2n-1)-bit product vector is first obtained $(S_{n-1},\ldots, S_1, C_{n-1},\ldots,C_1,C_o)$. Subsequently $S_{n-1},\ldots,S_1$ are reduced modulo the primitive polynomial $p(\alpha)$, specified by coefficients $f = (f_{n-1},\ldots,f_1,f_o)$. The hardware required is of the order $O(n^2)$ and it takes (n-1).t time delays, where t = total delay for one AND and Ex-OR gates (Fig. 6.6). Initially the a and b registers are loaded with α and then a series of 'Square', 'Multiply' operations are performed using the multiplier. An all 0s detector overrides the exponentiator output. (Fig. 6.7 and 6.8). To carryout one exponentiation a maximum of 64 multiplications are to be executed. Each multiplication takes a worst

case time of $\simeq$ 2 μ sec. using LSTTL. Hence a total of 64x2x4 μ sec. are required to carryout 4 rounds of the cipher transformation.

Key is moved into the key Registers, Plaintext data is moved into the Plaintext Register, and the ciphertext is loaded at the end of each transformation and moved out bytewise serially. (Fig. 6.7 and 6.9). The Plaintext is loaded into Left and Right Registers and the cipher transformation is carried out in parallel with I/O operations. This pipeline architecture provides a good speed of 125 k bits for LSTTL version and 250 k bits for STTL version. (Fig. 6.10).

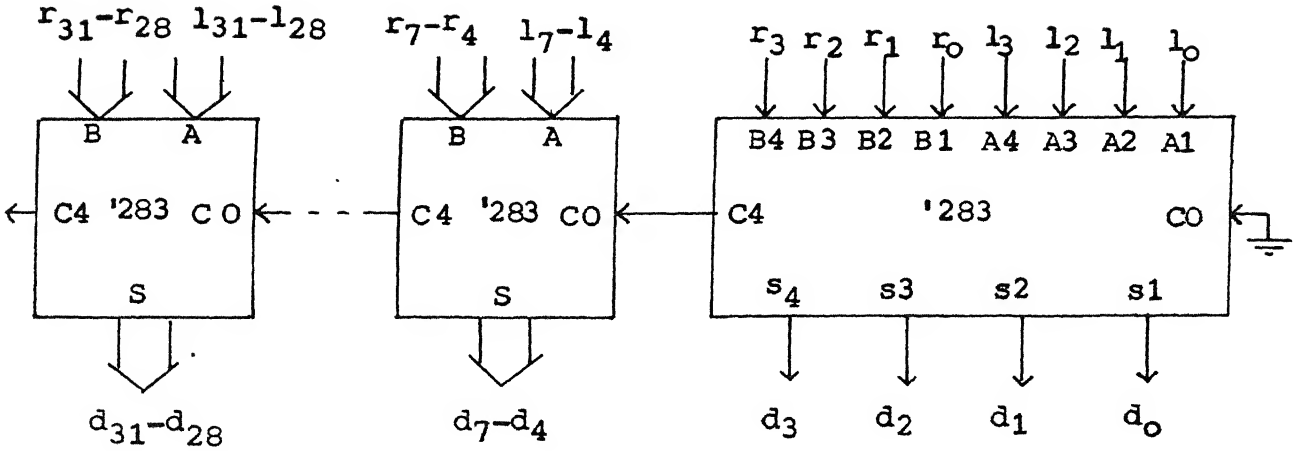
$$L = (l_{31}\ldots l_1 l_o)$$

$$R = (r_{31}\ldots r_1 r_o)$$

$$D = (d_{31}\ldots d_1 d_o)$$

$$D = L + R$$

$$\begin{array}{l} l_{31}\ldots l_1 l_o + \\ \underline{r_{31}\ldots r_1 r_o} \\ d_{31}\ldots d_1 d_o \end{array}$$



'LS283 4-bit Full Adder Specifications:

Max. delay from CO to s_i : 24 nsec

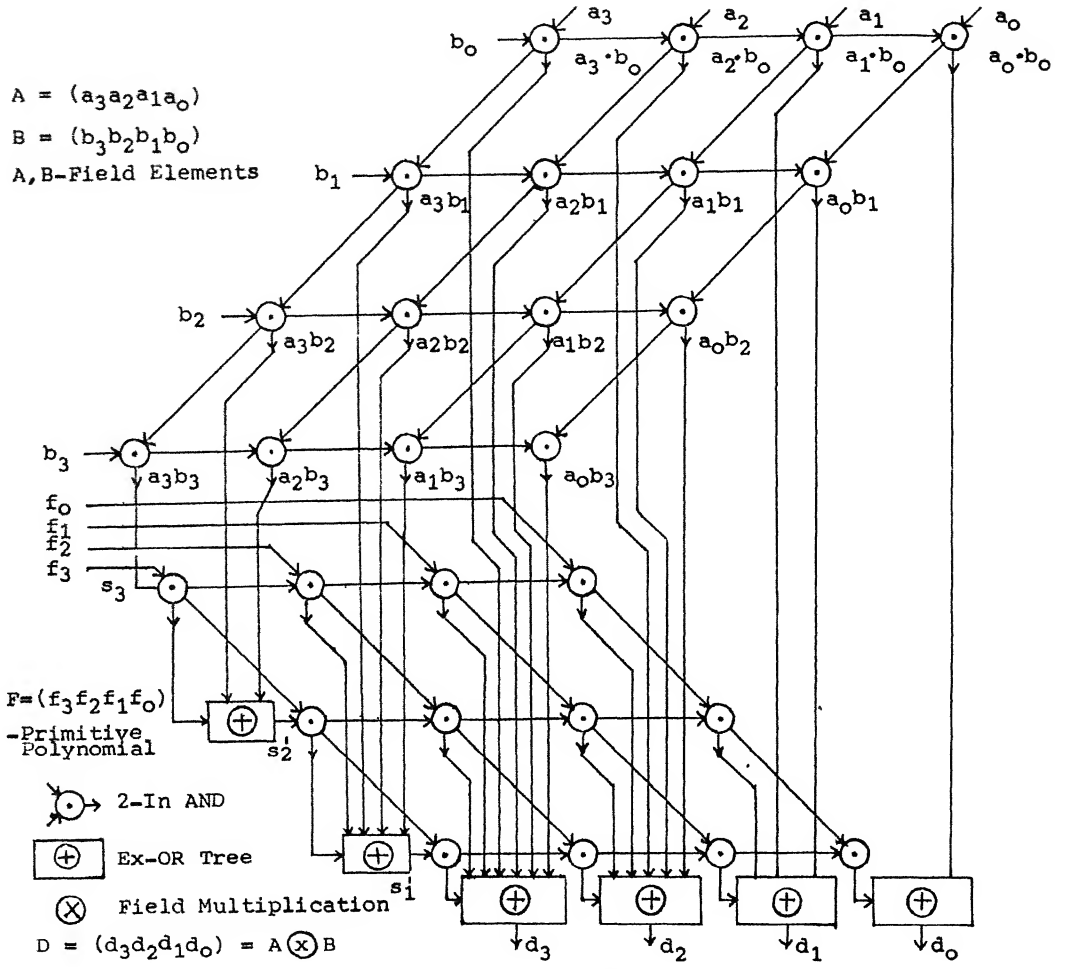
" CO to C4 : 17 nsec

" A_i or B_i to s_i : 24 nsec

" A_i or B_i to C4 : 17 nsec

Worst case addition delay (32 bit words) : 143 nsec

Fig. 6.5. Addition Modulo 2^{32}.



Specifications :

'LS280	9-bit Parity Generator	50 nsec max.
'LS08	Quad 2-In AND	20 nsec max.
'LS86	Quad 2-In Ex-OR	30 nsec max.

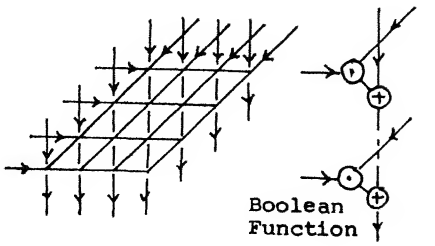
Worst case Field Multiplication delay (n = 32 bits)

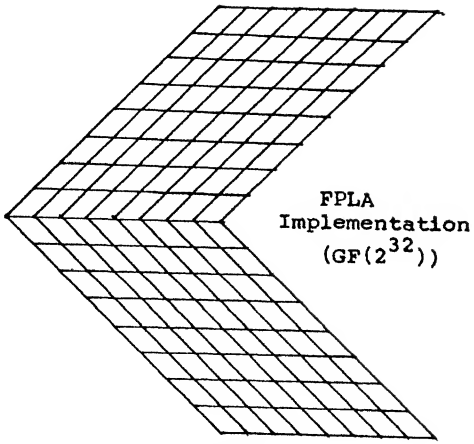
= (n-1).(Sum of AND and Ex-OR gate delays)

= 31x50nsec = 1.55 usec.

Fig. 6.6 GF(2^4) Array Multiplier.

825100 - 16x8 Field Programmable Logic Array (FPLA). Delay 50 nsec max.





Number of FPLAs 128

Worst case Multiply time ≃ 16x50 nsec = 0.8 usec.

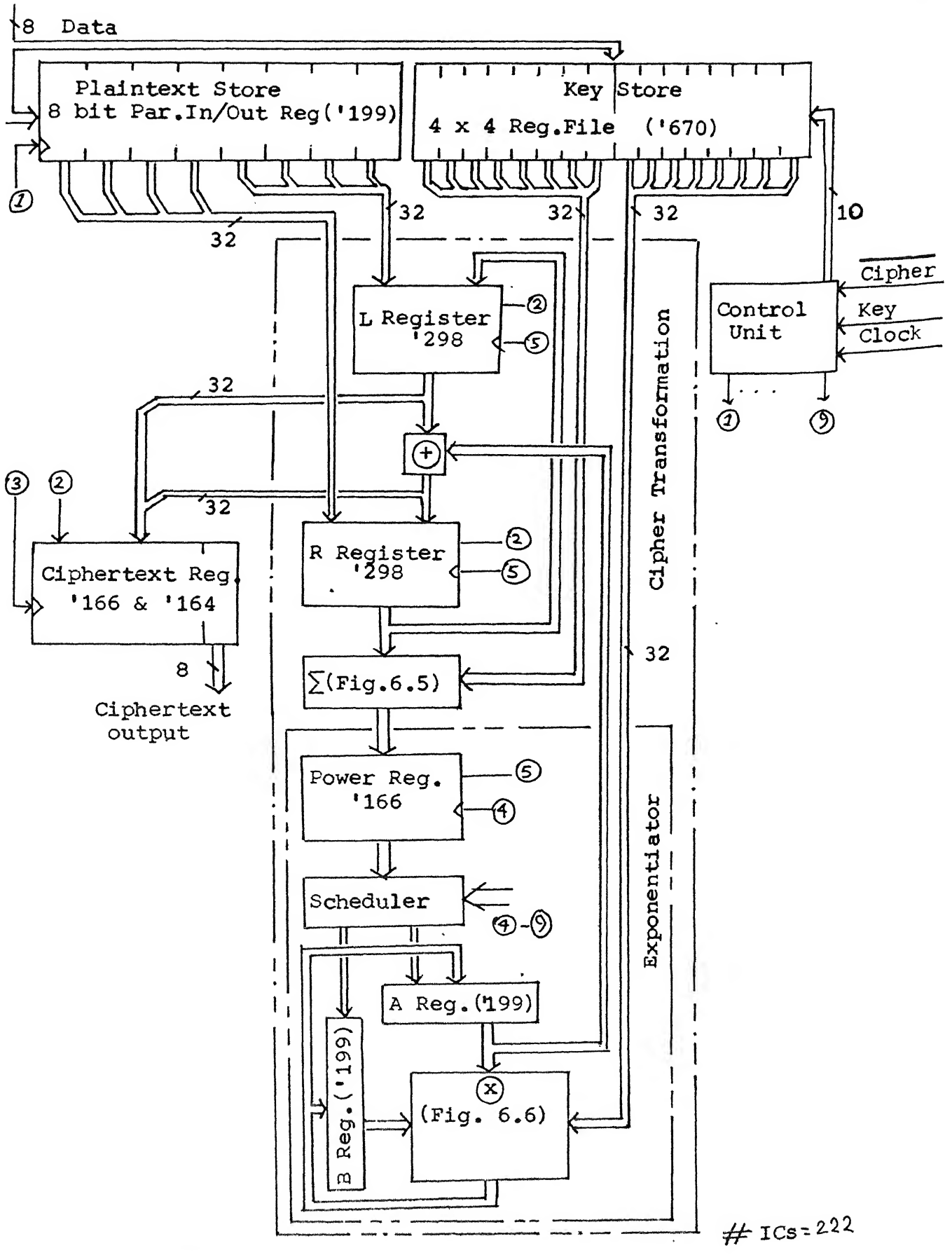
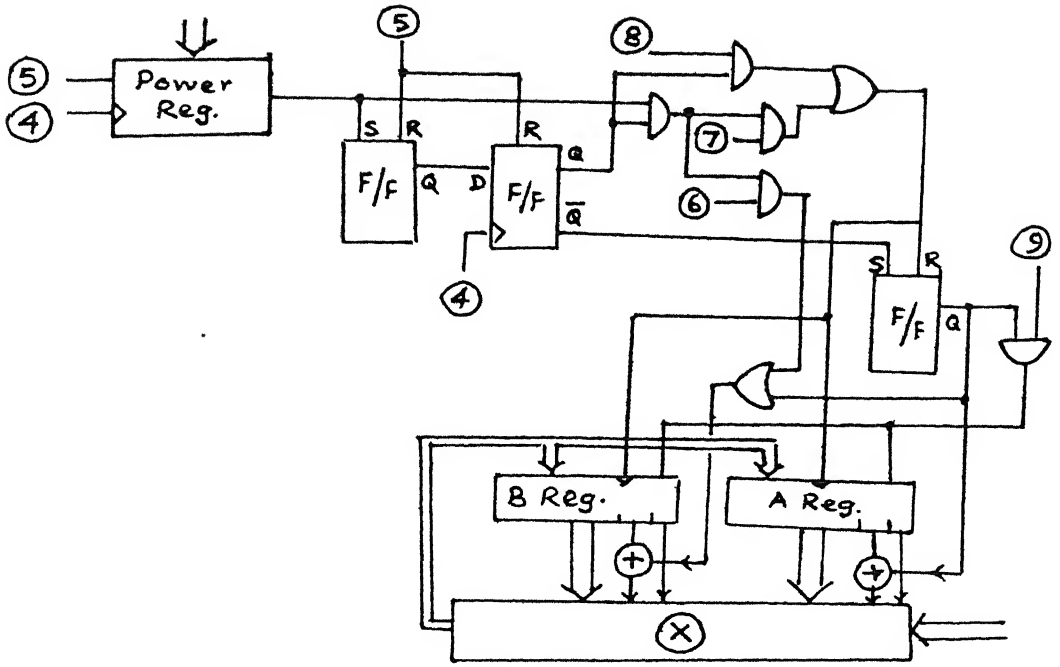


Fig. 6.7. Circuit Schematic



Exponentiator.

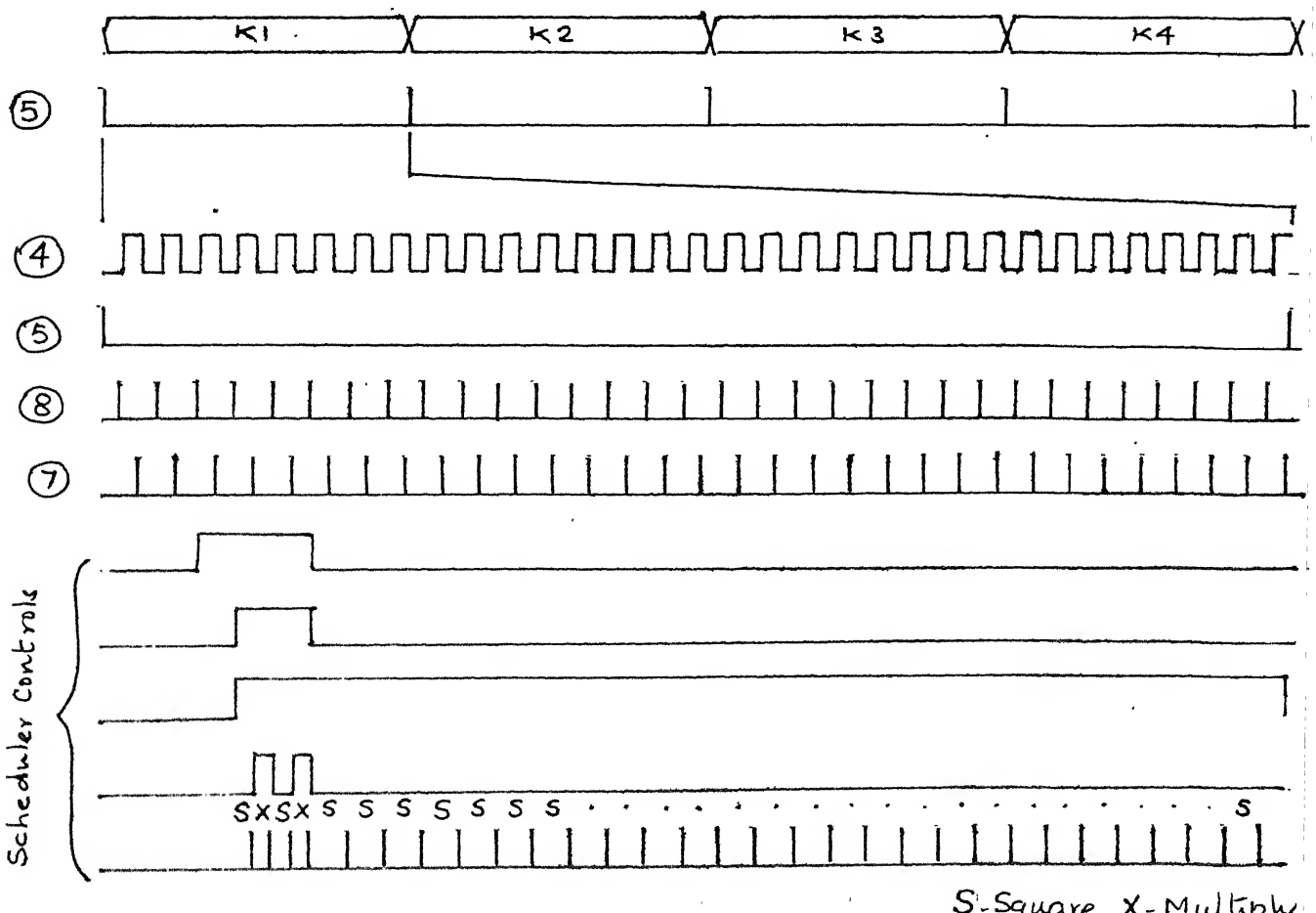
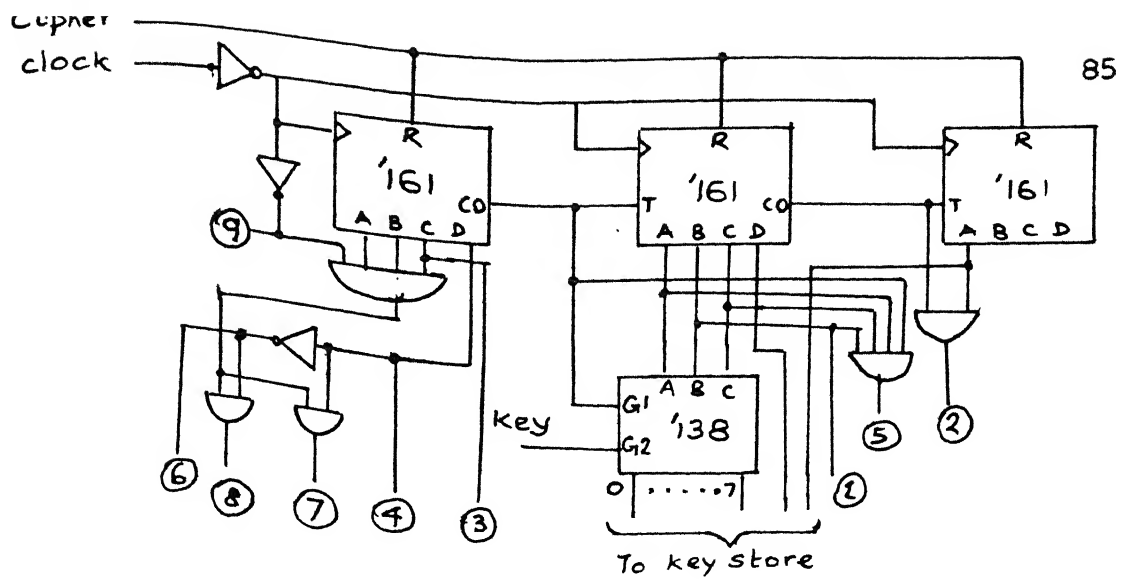


Fig. 6.8. GF(2^{32}) Exponentiator Timing Diagram



Control Unit

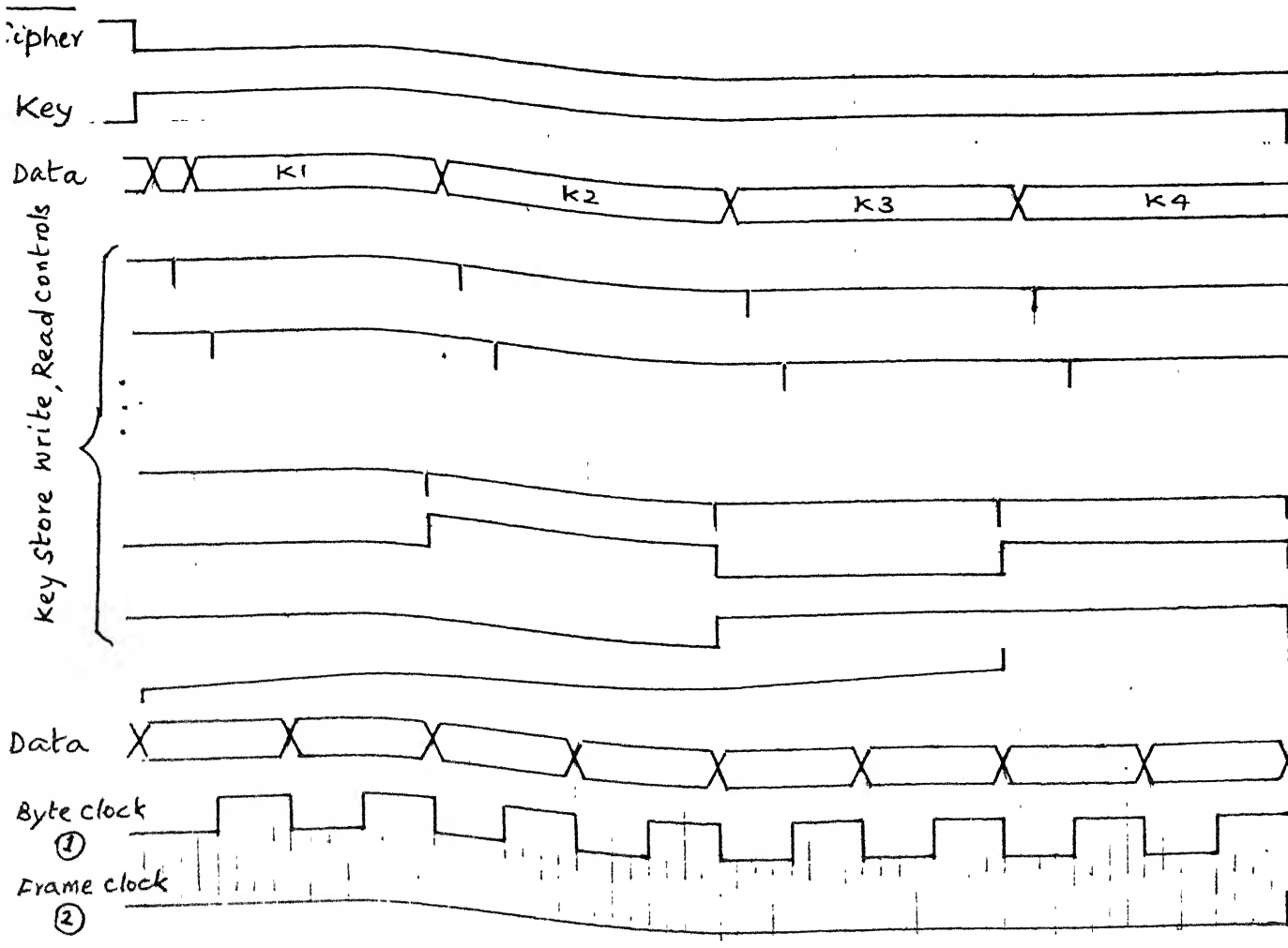
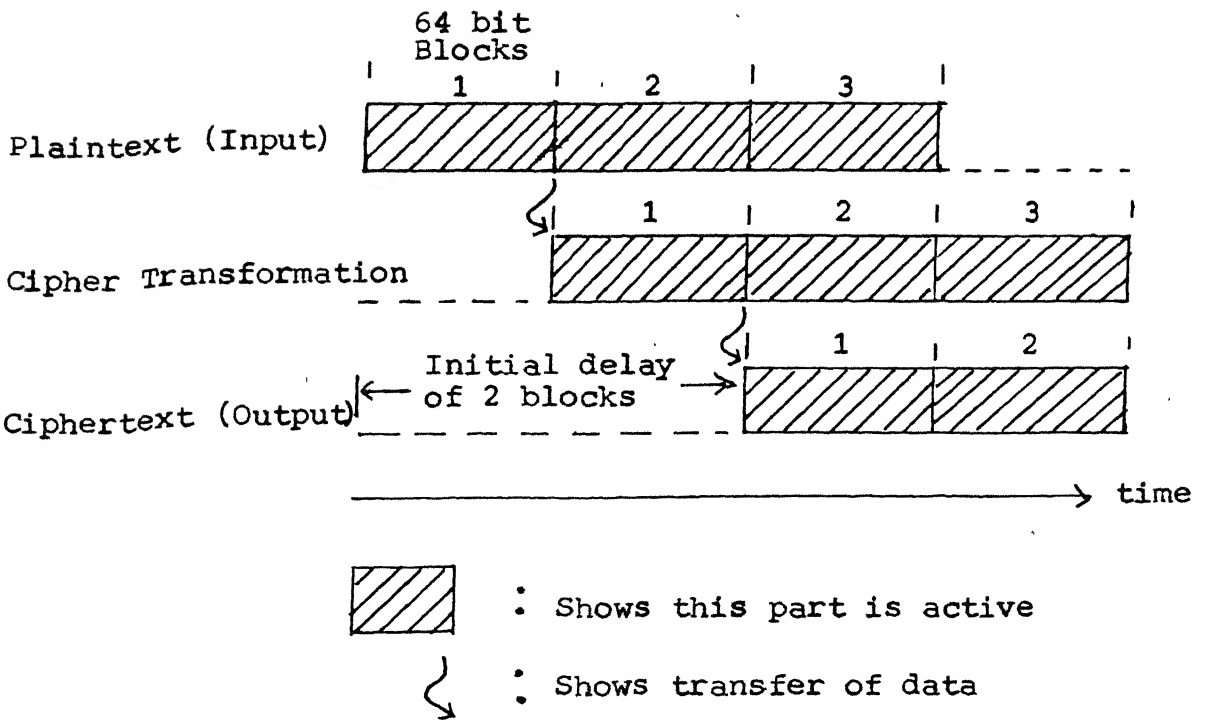


Fig. 6.9 T/C



Each Multiplication in $GF(2^{32})$ takes $\simeq$ 2 usec.

Each Exponentiation takes 64 Multiplications in $GF(2^{32})$ $\simeq$ 128 usec

Cipher Transformation on a block of 64 bits requires 4 rounds of exponentiation in $GF(2^{32})$ $\simeq$ 128x4 = 512 usec.

$$\text{Throughput} = \frac{64}{512 \times 10^{-6}} = 125 \text{ kbits (LSTTL version)}$$

$$= 250 \text{ kbits (STTL version)}$$

Fig. 6.10. Pipeline Operation.

CHAPTER 7

CONCLUSION

As permutations from the symmetric group S_N, on $N = 2^n$ members of a n bit binary vector, chosen randomly provides a strong cipher system, a hardware realisation is attempted. Elegant methods of mapping integer x ($0 \leq x \leq N!-1$) into permutation are presented and these offered very simple set up algorithm for a hardware permutation network. However the hardware complexity of these schemes are shown to be too high. As the number of permutations $2^n!$ of S_N is far too high than the size of the key space one could handle, quasi-random permutations out of S_N were considered as candidates for the cipher system. A way to choose these out of S_N using white binary sequences of 2^n length is postulated. It is shown using cyclic difference set concepts that it is possible to obtain these 2^n length sequences from the well studied (2^n-1) sequences obtained through $(2^n-1, 2^{n-1}, 2^{n-2})$-cyclic difference sets. Generalising the result lead to Galois Field $GF(2^n)$ transformation as the heart of the cipher system. This mapped the integers 0 to 2^n-1 (represented by n- tuple plaintext) to n-tuple representation of Galois Field elements $\alpha^{\infty}, \alpha^{0}, \ldots \alpha^{2^n-2}$. The hard problem of taking logarithm in $GF(2^n)$ is circumvented by the use of 'involution' operation. The resultant cipher system, having a

large key space and cryptographic strength, is shown to be realisable in hardware of reasonable complexity. Further the result indicates that the system can be configured for any value of n, thus putting no restrictions and if need be the design is extendable to higher values of 'n'. A particular hardware realisation of the same is proposed in detail for n = 64 using presently available MSI/LSI chips. A low power Schottky TTL version is capable of operating at 125 k bits/sec and a Schottky TTL version is capable of reaching 250 k bits/sec.

7.1 Results

. Cryptographically good permutation tables can be obtained through Galois Field GF(2^n) computation.

. Hardware and time complexity of the proposed final cipher system is polynomial and is capable of operating at 250 k bits/sec using STTL ICs.

7.2 Scope for Further Work

Concerted cryptanalytic attack on the proposed system could be carried out as future work. This will highlight the strength and weakness, if there are any, of the system. These might pave way to a polynomial time solution to cryptanalysis, in which case the cipher system is proved useless.

Only 'acceptable' solutions were obtained extending the results of already existing cyclic difference set results.

Theoretical solutions to near-ideal 2^n length sequences could be pursued. If this leads to results generalisable for all value of n, then these are better solutions to the problem. However on getting the solution one has to explore the possibility of translating the algorithm into a reasonable sized hardware.

The cipher system was derived through cyclic difference sets. This implied that cyclic shifts are to be used as the key space and necessiates the use of carry look ahead adder as a component of the system. If one could get a similar solution based on dyadic difference sets, then this will lead to faster and simpler hardware. The difficulty one faces here is that dyadic systems are defined only for values which are powers of 2 and hence no such approach as extending results of (2^n-1) length designs are possible as they don't exist.

Integrated circuit technology is taking strides and one could think of 10^6 devices in a silicon chip. A VLSI realisation of the hardware could be attempted. However one might have to modify the detailed implementation presented here to one that can be laid out with ease.

REFERENCES

[1] M. Abramowitz and I.A. Stegun, Handbook of Mathematical Functions, NY: Dover Publications, 1965.

[2] L. Adleman, "A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography", in Proc. IEEE 20th Annual Symposium on Foundations of Computer Science, 1979, pp. 55-60.

[3] T.C. Bartee and D.I. Schneider, "Computation with Finite Fields", Inform. contr., Vol. 6, pp. 79-98, Mar.1963.

[4] I.F. Blake, R.C. Mullin and S.A. Vanstone, "Computing Logarithms in $GF(2^n)$," in Lecture Notes in Computer Science, Vol. 196: Advances in Cryptology, G.R. Blakley and D. Chaum, Ed., NY: Springer-Verlag, 1985, pp.73-82.

[5] D. Coppersmith, "Fast Evaluation of Logarithms in Fields of Characteristic Two", IEEE Trans. Inform. Theory, Vol. IT-30, pp. 587-594, July 1984.

[6] D. Coppersmith and E. Grossman, "Generators for Certain Alternating Groups with Applications to Cryptography", SIAM J. Appl. Math., Vol. 29, No. 4, pp. 624-627, Dec. 1975.

[7] D.E. Denning, Cryptography and Data Security, Reading, MAS : Addison-Wesley, 1983.

[8] W. Diffie and M.E. Hellman, "Privacy and Authentication: An Introduction to Cryptography", Proc. IEEE, Vol. 67, No. 3, pp. 397-427, Mar. 1979.

[9] W. Diffie and M.E. Hellman, "New Directions in Cryptography", IEEE Trans. Inform. Theory, Vol. IT-22, No. 6, pp. 644-654, Nov. 1976.

[10] S. Even and O. Goldreich, "DES-like Functions can Generate the Alternating Group", IEEE Trans. Inform. Theory, Vol. IT-29, No. 6, pp.863-865, Nov. 1983.

[11] R. Fuji-Hara, I.F. Blake, R.C. Mullin and S.A. Vanstone, "Computing Logarithms in Finite Fields of Characteristic Two", SIAM J. Appl. Math., Vol. 29, No. 4, pp. 624-627, Dec. 1975.

[12] J.A. Gordon and H. Retkin, "Are Big S-boxes Best?" in Lecture Notes in Computer Science, Vol. 149: Cryptography, Thomas Beth, Ed., NY: Springer-Verlag, 1983, pp. 257-262.

[13] H. Gupta, Selected Topics in Number Theory, Kent, England: Abacus Press, 1980.

[14] J.K. Holmes, Coherent Spread Spectrum Systems, NY: John Wiley, 1982.

[15] V. Krishnamurthy, Combinatorics: Theory and Applications, New Delhi-Madras : EWP, 1985.

[16] D.E. Knuth, The Art of Computer Programming, Vol. II : Seminumerical Algorithms, Reading, MA : Addison-Wesley, 1981.

[17] A.G. Konheim, Cryptography : A Primer, NY: John Wiley, 1981.

[18] B.A. Laws and C.K. Rushforth, "A Cellular-Array Multiplier for GF(2^m)," IEEE Trans. Comput., Vol. C-20, pp. 1573-1578, Dec. 1971.

[19] F.J. MacWilliams and N.J.A. Sloane, "Pseudo-random Sequences and Arrays", Proc. IEEE, Vol. 64, pp. 1715-1729, 1976.

[20] W.W. Peterson, Error-Correcting Codes, Cambridge, MAS: MIT Press, 1961.

[21] S.C. Pohlig and M.E. Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance", IEEE Trans. Inform. Theory, Vol. IT-24, pp.106-110, 1978.

[22] R.L. Rivest, L. Adleman and M.L. Dertouzos, "On Data Banks and Privacy Homomorphisms", in Foundations of Secure Computation, R.A. Demillo, Ed., NY: Academic Press, 1978, pp.169-179.

[23] C.E. Shannon, "Communication Theory of Secrecy Systems", Bell Syst. Tech. J., Vol. 28, pp.656-715, Oct. 1949.

[24] N.J.A. Sloane, "Encrypting by Random Rotations", in Lecture Notes in Computer Science, Vol. 149: Cryptography, Thomas Beth, Ed., NY: Springer-Verlag, 1983, pp. 71-128.

[25] W.D. Wallis, A.P. Street and J.S. Wallis, Combinatorics: Room Squares, Sum-free Sets, Hadamard matrices, Lecture Notes in Mathematics, Vol. 292, NY: Springer-Verlag, 1972.

[26] C.C. Wang, T.K. Truong, H.M. Shao, L.J. Deutch, J.K. Omura and I.S. Reed," VLSI Architectures for Computing Multiplications and Inverses in GF(2^m)", IEEE Trans. Comput., Vol. C-34, pp.709-716, Aug. 1985.

[27] C.S. Yeh, I.S. Reed and T.K. Truong, "Systolic Multipliers for Finite Fields GF(2^m)", IEEE, Trans. Comput., Vol. C-33, pp.357-360, Apr. 1984.

[28] N. Zierler and J. Brillhart, "On Primitive Trinomials (Mod 2)", Inform. and Contr., Vol. 13, pp.541-554, 1968.

[29] N. Zierler and J. Brillhart, "On Primitive Trinomials (Mod 2)", Inform. and Contr., Vol. 14, pp.566-569, 1969.

[30] The TTL Data Book for Design Engineers, Texas Instruments, Texas: 1976.

[31] TTL Data Book, National Semiconductor Corporation, CA:1976.

List of Galois Field Elements and Difference Sets

Table A1.1 $GF(2^5)$

Power representation	5-tuple representation														
	$p(X)=1+X+X^2+X^3+X^5$					$p(X)=1+X+X^3+X^4+X^5$					$p(X)=1+X^3+X^5$				
	α^4	α^3	α^2	α	1	α^4	α^3	α^2	α	1	α^4	α^3	α^2	α	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
α^0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
α^1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
α^2	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
α^3	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
α^4	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
α^5	0	1	1	1	1	1	1	0	1	1	0	1	0	0	1
α^6	1	1	1	1	0	0	1	1	0	1	1	0	0	1	0
α^7	1	0	0	1	1	1	1	0	1	0	0	1	1	0	1
α^8	0	1	0	0	1	0	1	1	1	1	1	1	0	1	0
α^9	1	0	0	1	0	1	1	1	1	0	1	1	1	0	1
α^{10}	0	1	0	1	1	0	0	1	1	1	1	0	0	1	1
α^{11}	1	0	1	1	0	0	1	1	1	0	0	1	1	1	1
α^{12}	0	0	0	1	1	1	1	1	0	0	1	1	1	1	0
α^{13}	0	0	1	1	0	0	0	0	1	1	1	0	1	0	1
α^{14}	0	1	1	0	0	0	0	1	1	0	0	0	0	1	1
α^{15}	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0
α^{16}	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0
α^{17}	1	0	0	0	1	0	1	0	1	1	1	1	0	0	0
α^{18}	0	1	1	0	1	1	0	1	1	0	1	1	0	0	1
α^{19}	1	1	0	1	0	1	0	1	1	1	1	1	0	1	1
α^{20}	1	1	0	1	1	1	0	1	0	1	1	1	1	1	1
α^{21}	1	1	0	0	1	1	0	0	0	1	1	0	1	1	1
α^{22}	1	1	1	0	1	1	1	0	0	1	0	0	1	1	1
α^{23}	1	0	1	0	1	0	1	0	0	1	0	1	1	1	0
α^{24}	0	0	1	0	1	1	0	0	1	0	1	1	1	0	0
α^{25}	0	1	0	1	0	1	1	1	1	1	1	0	0	0	1
α^{26}	1	0	1	0	0	0	0	1	0	1	0	1	0	1	1
α^{27}	0	0	1	1	1	0	1	0	1	0	1	0	1	1	0
α^{28}	0	1	1	1	0	1	0	1	0	0	0	0	1	0	1
α^{29}	1	1	1	0	0	1	0	0	1	1	0	1	0	1	0
α^{30}	1	0	1	1	1	1	1	1	0	1	1	0	1	0	0

D = (4,5,7,9,12,16,18,19,20,21,22,24,25,28,29,30)
D = (4,6,7,9,11,15,16,17,19,20,21,22,23,26,29,30)
D = (4,6,8,9,10,12,13,17,18,19,20,21,24,25,27,30)
D = (1,6,8,10,11,12,14,15,19,20,21,22,23,26,27,29)

Table A1.2 $GF(2^6)$ primitive polynomial $p(X) = 1+X^5+X^6$

Power representation	6-tuple representation α^5	α^4	α^3	α^2	α	1	Power representation	6-tuple representation α^5	α^4	α^3	α^2	α	1
0	0	0	0	0	0	0	α^{31}	0	0	1	1	0	1
α^0	0	0	0	0	0	1	α^{32}	0	1	1	0	1	0
α^1	0	0	0	0	1	0	α^{33}	1	1	0	1	0	0
α^2	0	0	0	1	0	0	α^{34}	0	0	1	0	0	1
α^3	0	0	1	0	0	0	α^{35}	0	1	0	0	1	0
α^4	0	1	0	0	0	0	α^{36}	1	0	0	1	0	0
α^5	1	0	0	0	0	0	α^{37}	1	0	1	0	0	1
α^6	1	0	0	0	0	1	α^{38}	1	1	0	0	1	1
α^7	1	0	0	0	1	1	α^{39}	0	0	0	1	1	1
α^8	1	0	0	1	1	1	α^{40}	0	0	1	1	1	0
α^9	1	0	1	1	1	1	α^{41}	0	1	1	1	0	0
α^{10}	1	1	1	1	1	1	α^{42}	1	1	1	0	0	0
α^{11}	0	1	1	1	1	1	α^{43}	0	1	0	0	0	1
α^{12}	1	1	1	1	1	0	α^{44}	1	0	0	0	1	0
α^{13}	0	1	1	1	0	1	α^{45}	1	0	0	1	0	1
α^{14}	1	1	1	0	1	0	α^{46}	1	0	1	0	1	1
α^{15}	0	1	0	1	0	1	α^{47}	1	1	0	1	1	1
α^{16}	1	0	1	0	1	0	α^{48}	0	0	1	1	1	1
α^{17}	1	1	0	1	0	1	α^{49}	0	1	1	1	1	0
α^{18}	0	0	1	0	1	1	α^{50}	1	1	1	1	0	0
α^{19}	0	1	0	1	1	0	α^{51}	0	1	1	0	0	1
α^{20}	1	0	1	1	0	0	α^{52}	1	1	0	0	1	0
α^{21}	1	1	1	0	0	1	α^{53}	0	0	0	1	0	1
α^{22}	0	1	0	0	1	1	α^{54}	0	0	1	0	1	0
α^{23}	1	0	0	1	1	0	α^{55}	0	1	0	1	0	0
α^{24}	1	0	1	1	0	1	α^{56}	1	0	1	0	0	0
α^{25}	1	1	1	0	1	1	α^{57}	1	1	0	0	0	1
α^{26}	0	1	0	1	1	1	α^{58}	0	0	0	0	1	1
α^{27}	1	0	1	1	1	0	α^{59}	0	0	0	1	1	0
α^{28}	1	1	1	1	0	1	α^{60}	0	0	1	1	0	0
α^{29}	0	1	1	0	1	1	α^{61}	0	1	1	0	0	0
α^{30}	1	1	0	1	1	0	α^{62}	1	1	0	0	0	0

D = (5,6,7,8,9,10,12,14,16,17,20,21,23,24,25,27,28,30
33,36,37,38,42,44,45,46,47,50,52,56,57,62)